LIFE-LONG MAPPING OF OBJECTS AND PLACES IN DOMESTIC ENVIRONMENTS

A Thesis
Presented to
The Academic Faculty

by

John G. Rogers III

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in
Robotics

Robotics and Intelligent Machines
Georgia Institute of Technology
May 2013

LIFE-LONG MAPPING OF OBJECTS AND PLACES IN DOMESTIC ENVIRONMENTS

Approved by:

Professor Henrik I. Christensen,
Advisor
College of Computing
Georgia Institute of Technology

Professor Frank Dellaert
College of Computing
Georgia Institute of Technology

Professor Tucker Balch
College of Computing
Georgia Institute of Technology

Professor Ayanna Howard
Electrical and Computer Engineering
Georgia Institute of Technology

Professor Kostas Daniilidis
Department of Computer and
Information Science
University of Pennsylvania

Date Approved: 20 December 2012

To my wife Abbey

and my parents John and Rosa,

whose love and support made this work possible;

and to Tianyu,

who is lost but not forgotten...

ACKNOWLEDGEMENTS

Many of the contributions in this thesis were made in collaboration with my colleagues at Georgia Tech: Alexander J.B. Trevor, Carlos P. Nieto-Granda, and my advisor Henrik I. Christensen, as well as Professor Frank Dellaert and his students Richard Roberts and Alex Cunningham.

I would like to thank the ARL MAST CTA for providing me with the funding and opportunity to work with some great scientists across the country. Our early multi-robot experiments used some components which were developed at JPL by Dr. Jeremy Ma and Dr. Larry Matthies. I would like to thank Professor Vijay Kumar, Professor Kostas Daniilidis, and Professor Nathan Michael for inviting me to the GRASP lab at the University of Pennsylvania to use their robots and facilities for some of the multi-robot experiments in this thesis. I would also like to thank the students at the GRASP lab for helping me use their equipment and writing framework code and drivers used on their robots, including Jon Fink, Daniel Mellinger, and Shojie (Frank) Shen.

I had the unique opportunity to work with the scientists at the Army Research Lab Computational and Information Sciences Directorate. Stuart Young invited me to work with their group for the past two summers. I would like to thank Stuart for the opportunity he provided me to implement my algorithms on his military robots, and for bringing me to several field experiments at military training facilities. I would like to thank ARL scientists Dr. Ethan Stump, Dr. Jon Fink, Dave Baran and Jason Gregory for their assistance in running experiments, helpful discussions, and getting robots to work.

Finally, I would like to thank my committee for reviewing my work and their

helpful suggestions which enabled this thesis, including my thesis advisor Professor Henrik Christensen, Professor Frank Dellaert, Professor Ayanna Howard, and Professor Tucker Balch. I would also like to especially thank my external committee member Professor Kostas Daniilidis from the University of Pennsylvania.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In the future, robots will expand from industrial and research applications to the home. Domestic service robots will work in the home to perform useful tasks such as object retrieval, cleaning, organization, and security. The tireless support of these systems will not only enable able bodied people to avoid mundane chores; they will also enable the elderly to remain independent from institutional care by providing service, safety, and companionship. Robots will need to understand the relationship between objects and their environments to perform some of these tasks. Structured indoor environments are organized according to architectural guidelines and convenience for their residents. Utilizing this information makes it possible to predict the location of objects. Conversely, one can also predict the function of a room from the detection of a few objects within a given space.

This thesis introduces a framework for combining object permanence and context called the *probabilistic cognitive model*. This framework combines reasoning about spatial extent of places and the identity of objects and their relationships to one another and to the locations where they appear. This type of reasoning takes into account the context in which objects appear to determine their identity and purpose. The *probabilistic cognitive model* combines a mapping system called *OmniMapper* with a conditional random field probabilistic model for context representation. The conditional random field models the dependencies between location and identity in a real-world domestic environment. This model is used by mobile robot systems to predict the effects of their actions during autonomous object search tasks in unknown environments.

I

INTRODUCTION

Robotics and automation have revolutionized industry to provide mankind with inexpensive, high quality consumer products which make modern society possible. Automation currently operates within the home to perform simple jobs like washing dishes and vacuuming the floor; however, in the near future, the domestic roles and responsibilities given to robots will expand. Domestic service robots will work in the home to perform useful tasks such as object retrieval, cleaning and organization, and security. The tireless support of these systems will not only enable able bodied people to avoid mundane chores; they will also enable the elderly to remain independent from institutional care by providing service, safety, and companionship. Despite significant automation already being present in the home such as dishwashers, washing machines, and robot vacuums, people face a steadily increasing amount of duties necessary to support their current lifestyles. In the future, domestic service robots will take care of our chores so we will have more time for cultural and intellectual pursuits; these robot servants will need to be able to understand their environments to enable this type of capability.

There is still a long way to go before this future becomes a reality; however, to date significant progress has been made. Robots have been developed which assist the elderly and infirm in performing fetch-and-carry tasks using human guidance and feedback, as with the robot *EL-E* in Nguyen et al. [2008]. The introduction of the PR2 program has provided sophisticated robot hardware to research teams so they can focus on developing software to increase the capability of robots in service roles. This program has developed a robot system which can navigate perpetually in an office

environment while recharging itself when its batteries become depleted in Meeussen et al. [2011]. Dozens of these robots have been provided to researchers around the world; this has resulted in some interesting new robot capabilities such as folding laundry [Miller et al., 2011], manipulating generic objects in the home [Ciocarlie et al., 2010], and in human-robot interaction [Goodfellow et al., 2010]. The PR2 robot platform is physically capable of handling many tasks in the home; however, it has a limited payload capacity, it has poor manual dexterity, and it has limited range of motion. Before a domestic service robot can be built as a viable commercial product, additional challenges must be addressed. A domestic service robot must have sufficient dexterity to manipulate a diverse set of objects. Mobility, especially in cluttered environments, is still a challenge.

There are a number of requirements which must be considered when designing systems that interact with the environment in more sophisticated ways than navigation from one position to another.

Traditionally, maps have been organized entirely as a collection of interrelated geometric features. Structured indoor environments are organized according to a number of rules. These rules include:

- architectural design rules
- organization of spaces according to function
- organization of objects within spaces according to application

Utilizing this information makes it possible to predict the location of objects, which enables an efficient search procedure for objects, places, and rooms. Conversely, one can also predict the function of a room from the detection of a few objects within a given space.

To facilitate the use of such information, there is a need to organize the knowledge about objects and places into a coherent representation. This allows a robot to use

it to represent a particular space and to reason about objects, contexts, and spaces to perform efficient searches. To bring about such a representation, there is a need to consider:

- Recognition of a broad set of objects
- Representation of spatial relationships
- Utilization and integration of a diverse set of features

Clearly, it is also of interest to demonstrate that such a representation can be used to implement systems that are efficient in terms of representation of space and retrieval of objects.

A global representation for the location and identity of objects in a domestic environment can be used by a robot to perform a variety of service tasks such as fetch-and-carry, retrieval, cleaning, and meal preparation. Objects are the critical element with which domestic service robots will have to interact; therefore, a representation which tracks the location of objects is needed. The objects within a home are used for a variety of such tasks and are required to perform many operations which are expected of robots. Human environments are organized and segmented according to purpose; the objects used for these purposes will tend to be found in the rooms where their associated tasks will be performed. For example, the pots, pans, spoons, and forks will be found in the kitchen, where they are used for meal preparation. Electronic gadgets such as tablet computers, stereos, and televisions are usually found in an office setting or a living room.

One of the most important tasks that a new domestic service robot must be capable of is the first one that it will perform when it is unpacked from its shipping crate: mapping and familiarizing itself with its new home. This is the specific task: the robot will autonomously build a map of its environment and deduce the purpose of each room and learn where some objects are with which it might be required to

interact in the future. The robot could also ask the user to give it a *tour* of the user's home. During this *tour*, the user could point out and identify relevant places and objects that the robot will need to perform its duties. The commercial version of the mapping and familiarizing task will probably leverage both automatic and human-interactive components; however, for the purpose of this thesis we will focus on the automatic components.

The position of an object within the environment can be used as a cue for that object's identity. For example, the microwave oven is more likely to be found in the kitchen, and the toilet is more likely to be found in the bathroom. The knowledge of the purpose of the room currently inhabited by the robot can be used to help the robot identify other objects in the room. The recognition of some objects can also be used as a cue for context of where to find other related objects around them, such as the mouse is usually to be found to the right of the keyboard, or the light switch should be found on the knob side of the doorway.

The task of identifying objects in an unknown (and dynamic) environment should incorporate spatial location and object permanence. In this way, object recognition and *simultaneous localization and mapping*(SLAM) are linked; the performance of each is improved by the other. Object recognition can provide a strong cue for data association, and spatial position can provide a strong cue for object identification. Prior identification of an object from a certain vantage point, combined with *object permanence*, the expectation that things remain where they were last seen for short periods of time, can be used to simplify the future recognition task by limiting the search space and permitting less certain matches to be incorporated if they agree with previous measurements of the object.

1.1 Problem Studied

This thesis introduces a framework for combining object permanence and context called the *probabilistic cognitive model*. This framework combines reasoning about spatial extent of places and the identity of objects and their relationships to one another and to the locations where they appear. This type of reasoning takes into account the context in which objects appear to determine their identity and purpose. The *probabilistic cognitive model* combines a mapping system called *OmniMapper* with a conditional random field probabilistic model for context representation. The conditional random field models the dependencies between location and identity in a real-world domestic environment. Several studies will be presented which address various aspects of mobile robot mapping. Experimental results are presented for simulated environments in addition to two unique real-world live robot scenarios. The techniques described in this thesis were trained with data from various sources and tested in real-world environments.

The problem to be addressed in this work is: how can a mobile robot system leverage contextual relationships in structured indoor environments to improve performance on an object search task? This problem can be broken down into four sub-problems which are:

- How can a robot segment and classify objects and places?
- How can a model of the contextual relationships between objects and places be used by a robot to perform tasks?
- How can contextual relationships between objects and places be learned from training data?
- How can the contextual model be updated and adapted during online operation?

The first sub-problem involves two aspects: segmentation and classification. Places

should be segmented according to architectural boundaries corresponding to rooms. Objects should be segmented from the background to isolate them for classification. The second aspect of this sub-problem is classification of objects and places. Classification provides information about known entities which can be used to make inference about unknown entities through context.

The second and third sub-problems involve building and using a model of contextual relationships between objects and places. This model of contextual relationships, called the *probabilistic cognitive model*, allows a mobile robot to make inference about unknown entities through other recognized entities. This inference takes place via a contextual model which is learned from training data.

The final sub-problem involves adapting to unexpected circumstances. A mobile robot system operating in an unknown and unfamiliar environment may be assisted by its models built from training data; however, these models may be insufficient for all situations. A versatile robot system should be able to update these models based upon its experiences to master its environment.

1.2 Contributions

There are four primary contributions presented in this thesis.

- The first contribution is a novel representation for expressing the contextual relationship between objects and places called the *probabilistic cognitive model*(PCM).

- The second contribution is a methodology for using the PCM in a planner to estimate the result of executing actions in an unknown environment.

- The third contribution is an evaluation of the PCM and the planner in experiments; the experiments consist of a demonstration in simulation which shows that the use of learned contextual relationships accelerates an object search task over an uninformed strategy. The system is then evaluated in two live-robot

applications: a domestic service robot searching for household goods, and a military robot searching for WMDs in a simulated insurgent stronghold.

- The fourth contribution presented in this thesis is a methodology for performing metric and topological mapping called *OmniMapper*.

1.3 Outline of the thesis

Chapter 2 will discuss the advances we have made on the field of mobile robot simultaneous localization and mapping (SLAM). This chapter will present algorithms which support performing SLAM in domestic environments. A notorious problem in mapping is the management of complexity. As more information is gathered from the environment, the robot's model becomes more complex. To manage this complexity, there is a need to partition the map representation into a set of sub-maps that are topologically connected. The first part of this chapter describes a method for automatic partitioning of maps into sub-maps which correspond to logical room boundaries. The second part of this chapter develops an algorithm for correcting errors in mapping caused by moving objects: a problem which is critical to provide life-long mapping. This chapter then describes the development and capabilities of the mapping library called *OmniMapper*, as well as its applications, modules, and extensions. Finally, a study is presented which addresses the cost and complexity of mobile robots by analyzing what is truly necessary in terms of sensor accuracy to provide robot mapping performance.

In the future, it is likely that multiple robots will be deployed in homes and in factory settings. Integration of mapping across multiple platforms is thus an important problem. **Chapter 3** will discuss how the *OmniMapper* library has been extended to work across teams of robots to perform multi-robot mapping. The first part of this chapter will describe a technique for sharing information across a small team of robots as well as a strategy for recruiting teammates to explore branching structures

such as intersections in a hallway. A cost-effective means of equipping a team of simple robots is to provide the team members with heterogenous sensor suites; a few robots are given sophisticated sensors and the rest compliment them with inexpensive sensors. The second part looks at small teams of mobile robots with heterogenous sensors, one is 2D and the other is 3D. The final section of this chapter looks at how larger teams of up to 10 robots can coordinate their efforts to explore an unknown environment efficiently.

Features are recognized when the robot sees them again by their location in the environment and are associated with the landmarks which have been mapped previously. Sometimes, this association process will fail due to perceptual aliasing, insufficient constraints, or if the robot is lost. *Object recognition* can provide *semantic* data association cues beyond geometric location. **Chapter 4** opens with a study which compared two popular visual features for their performance in object recognition and classification tasks. The second section describes a machine learning technique which was developed for detecting strong visual cues in the environment. This algorithm was then used to detect and map door signs in an office environment, together with a technique for reasoning about what the semantic content of the cue indicated for the purpose of performing data association. Finally, the new software components for object recognition and classification are presented.

Chapter 5 will describe a new technique for representing context between places and objects in semantic SLAM called the *probabilistic cognitive model*. This technique couples object recognition and mapping via a graphical model which is trained to represent the context of objects and rooms. The experiments in this chapter are performed with a tele-operated robot, where the probabilistic cognitive model is passively incorporating information to determine the identities of rooms and other objects based on what it has seen.

Chapter 6 will demonstrate how the *probabilistic cognitive model* from chapter 5

can be utilized for enabling autonomous robot planning on an object search task. The probabilistic cognitive model from chapter 5 is used to predict the *world state* which will result from the robot's actions. The PCM-Planner uses this ability to predict the results of actions to select which action will help it find a target object. This system is tested in simulation as well as on two types of robot platforms and in two scenarios, a domestic service robot scenario and a military counter-insurgency scenario.

Chapter 7 will present discussion regarding how these components and studies support this thesis. A review of key experiments is presented with a discussion of how these experiments relate to one-another and how they can be extended and merged.

II

MOBILE ROBOT MAPPING

Maps were developed to allow agents to navigate in a region without getting lost. One of the earliest maps of the world was made by Anaximander in the 5th century BC. This map is very approximate and enables costal navigation of the Mediterranean and the Black Sea to reach areas in Asia, Europe, and Africa. Since this time, cartographers have improved map making through technology such as the compass, the sextant for measuring latitude, clocks for establishing longitude. Now, we build accurate maps using centimeter accuracy through GPS and satellite imagery.

These types of maps are useful for navigating ships at sea and along highways in the countryside, but are not very useful for robots operating indoors. Robots use various kinds of maps to navigate indoors including:

- Occupancy grids
- Landmark maps
- Topological maps
- Graphical models
- Semantic maps

Occupancy grids and topological maps are used by robots to plan trajectories to navigate across distances outside their sensor horizon. Graphical models and semantic maps incorporate additional information about the environment that robots can use to accomplish their goals. This work is primarily about using semantic maps and graphical models for robots to perform tasks; however, occupancy grids and topological maps are also used for obstacle avoidance and navigation.

To return to a particular location, there is a need for geometric maps which allow navigation and localization with respect to landmarks in the environment. Landmark-based mapping is challenged by a number of issues, such as:

- Computational complexity
- Data association errors
- Static and dynamic landmark assumptions
- Integration of a diverse set of features

We will address each of these issues and establish a basis for the *probabilistic cognitive model* in this chapter. Background information on robot mapping will be given in section 2.1.

This chapter discusses some initial developments which enable mobile robots to segment independent components by analyzing the information which is shared across rooms. This enables mobile robot mapping to maintain constant-time updates to a small local map which corresponds to a logical partition in a domestic environment such as rooms. The study is detailed in section 2.2.

The second component described in this chapter in section 2.3 is a study on how an expectation-maximization algorithm can segment bad data associations which come from an object which has been moved during mapping. This section also introduces a simple version of object SLAM with high level data association determined by AR Toolkit (QR code-like) markers. The EM algorithm can correct map error caused by assuming that these moveable objects are static and partition the set of landmarks into a dynamic and static set, effectively detecting which objects are fixed and which are likely to be moved. This algorithm is related to current state-of-the-art techniques for removing erroneous data associations in large-scale city wide mapping efforts.

After these two algorithms were developed, we focused on producing a mapping library which is capable of handling multiple sensor modalities and feature as well

as feature-less representations of landmarks in the environment. This system, called *OmniMapper* is described in section 2.4 along with its software plugins and infrastructure which makes the system extensible.

Mobile robots in the laboratory are often very sophisticated and expensive; these aspects are highly restrictive for their adoption in the marketplace. In addition, the expense associated with mobile robots might restrict their application in multi robot scenarios. We performed a study which used the *OmniMapper* on a robot with a variety of sensor configurations to determine where simpler and less expensive components could be substituted. This study is described in section 2.5.

2.1 Background

Simultaneous Localization and Mapping(SLAM) has been an active research topic since the mid 1980s. The problem is the synthesis of two simpler problems: localization in a known map, and map building with known location. Currently, many researchers believe that SLAM is a solved problem; however, few plug-in systems exist and there are few commercial applications in the marketplace today.

Smith and Cheeseman [1986] presented an extension of the extended Kalman filter(EKF) used for mobile robot localization. The EKF is augmented with landmark state in addition to robot state and both are updated simultaneously. This has the advantage of simplicity and constant-time operation on a fixed size world. Since past robot pose information is marginalized out at each update step, linearization errors are irreversible and will accumulate and lead to overconfidence and eventual failure. In addition, any exact implementation of this type will require the formation and inversion of a dense covariance matrix. The standard EKF formulation requires the inversion of the covariance matrix in the update step; a more advanced representation is in the information form. The information matrix representation requires inversion to perform data association. Current best algorithms for matrix inversion exploiting

the symmetric positive definiteness property of a covariance matrix involve $O(n^2)$ steps; this is a serious limitation which precludes large scale implementation. More details about the early approaches to the SLAM problem can be found in Durrant-Whyte and Bailey [2006] and modern approaches in Bailey and Durrant-Whyte [2006].

The second type of SLAM back-end is called *graph SLAM*. In the *graph SLAM* approach, all robot poses are maintained and optimized along with landmark positions. To the layperson, this might seem to be a mistake since the trajectory grows as the robot moves and could make the algorithm less efficient than the EKF algorithm. Paradoxically, the graph SLAM algorithms are actually much more efficient than EKF for large sized problems, and they also overcome linearization bias errors which plague EKF based algorithms. This is due to the fact that by not marginalizing out past poses as in the EKF, the graph SLAM algorithms maintain sparsity in the measurement matrix. EKF based algorithms will form a fully dense covariance matrix which must be inverted; however, graph based techniques will form a larger, sparse measurement matrix which can be efficiently solved via sparse QR factorization. Folkesson and Christensen developed GraphSLAM [Folkesson and Christensen, 2004], which was able to close loops and avoided linearization error through the use of a nonlinear optimization engine. Loop closure was achieved by adding human-guided constraints between features and then re-optimizing. Dellaert [2005] developed the Square Root SAM algorithm which uses sparse Cholesky factorization to optimize a set of landmark measurements and the robot trajectory in an efficient manner. Further progress has been made on online solutions to the SAM problem which uses QR factorization for reordering the measurements to get optimal and online or incremental updates such as with incremental SAM (iSAM) [Kaess et al., 2007, 2008].

2.2 SLAM with normalized graph cuts[1]

The original approach to the SLAM problem is to update a state vector and covariance matrix composed of the robot pose and the estimated landmark positions with the Extended Kalman Filter (EKF). This technique evolved out of the original successful application of the EKF for mobile robot localization with an *a priori* map in Crowley [1989] and Chatila and Laumond [1985]. By placing the estimated landmark locations within the state vector and updating them simultaneously, the first SLAM implementation was reported by Smith and Cheeseman [1987].

The graph SLAM and SAM algorithms described in section 2.1 represent progress towards a large-scale solution to the SLAM problem; however, each of them requires an increasing amount of computation per update step. EKF updates suffer from quadratic complexity per update step, iSAM updates can be held to near linear complexity per update with proper column reordering. Sub-mapping strategies have been developed to *approximate* the solution to the SLAM problem while maintaining *constant time* update steps. This is accomplished by restricting the capacity of the sub-map to a fixed maximum number of landmarks. The filter makes the approximation that other sub-maps are independent and performs updates on fixed sized sub-maps in constant time.

Sub-mapping strategies such as Atlas [Bosse et al., 2003] and Hierarchical SLAM [Estrada et al., 2005] maintain constant sized local maps and offer loop closure updates which are linear in the size of the loop, in terms of local map frames traversed. Other algorithms postpone global updates until they are needed and focus on local updates such as the Compressed EKF [Guivant and Nebot, 2001]. Techniques within the SAM community such as Tectonic SAM [Ni et al., 2007] are able to optimize sub-maps to generate exactly the same results as full Square Root SAM but with the efficiency of

[1] ***This section is based upon Rogers and Christensen [2009]***

a constant sized map. This is accomplished by performing a final map optimization on all measurements using the sub-map result as an initial condition. Since each local map starts with the robot perfectly localized in these schemes, sub-maps offer some protection against the data association ambiguities in nearest-neighbor gating which would otherwise be more difficult with larger pose uncertainties. Loop closure in sub-map algorithms requires joint data association such as from Neira and Tardós [2001]; however, pose uncertainties are small enough within a local map to permit nearest-neighbor gating. This is a significant advantage for the visual SLAM technique used in this section which relies on low uncertainties to achieve real-time performance while measuring visual features. These sub-mapping strategies make fixed, capacity-based decisions about when a sub-map should be made; this section will demonstrate an alternative strategy based on *normalized graph cuts* from the image segmentation community which will choose partitions between sub-maps which remove less information than arbitrary partitions, providing a better approximation to the full map.

This section presents a new strategy for choosing partitions in the Atlas framework based on normalized graph cuts. The software described in this section uses Andrew Davison's SceneLib[Davison, 2003, Davison and Murray, 2002, Davison, 1998][2]. The unified inverse-depth parameterization from Montiel et al. [2006] is used for landmark representation, and the Atlas framework of Bosse et al. [2003] is used to maintain local maps. These components will be described in section 2.2.1 with implementation details in section 2.2.2. Normalized graph cuts will be used to find locally optimal partitions to separate local maps in a simulated and live robot experiment in section 2.2.3 with discussion of conclusions in section 2.2.4.

[2]Andrew Davison has made the MonoSLAM software available at http://www.doc.ic.ac.uk/ ajd/Scene/index.html

2.2.1 Background

2.2.1.1 Monocular SLAM

Davison's work in [Davison, 2003, Davison and Murray, 2002, Davison, 1998] showed that it is possible to perform real-time SLAM with a monocular camera without the use of odometric or inertial motion feedback. Real-time performance is accomplished by projecting the covariance ellipse of a feature into the image; which, by taking the 99% confidence level yields a search region which can be examined to find the feature. Since this region is much smaller than the overall image, a template search can quickly yield an accurate measurement of the feature. This can be done at full frame-rate which allows for the use of a velocity motion model without any odometric or inertial feedback.

2.2.1.2 Inverse Depth

The software used in this section uses the core components of SceneLib, but the two-phase feature initialization is removed by using the unified inverse-depth parameterization [Montiel et al., 2006]. In the inverse depth parameterization, the feature state in the filter is expanded from the 3 spatial coordinates into a spherical coordinate system consisting of the origin of the feature, which was the robot position when it was first observed, the horizontal bearing θ, the vertical bearing ϕ, and the *inverse depth* ρ. The feature state vector is shown in equation 1.

$$y_i = \begin{pmatrix} x_i \\ y_i \\ z_i \\ \theta_i \\ \phi_i \\ \rho_i \end{pmatrix} \tag{1}$$

Using this parameterization, the projection of the feature into the camera is shown

in equation 2. This projection makes use of the inverse depth of the landmark, which allows it to be well-defined even when $\rho_i = 0$, when the depth is infinite. The Jacobians in SceneLib were re-computed with respect to the inverse depth representation to switch to this representation.

$$h^C = R^{CW}\left(\rho_i\left(\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} - r^{WC}\right) + \begin{pmatrix} \sin\theta_i \cos\phi_i \\ \cos\theta_i \cos\phi_i \\ \sin\phi_i \end{pmatrix}\right) \tag{2}$$

2.2.1.3 Atlas

The approach described in this section uses the Atlas framework developed by Bosse et al. [2003], which is a technique for sub-mapping based on computing the minimum uncertainty path to each other frame using a modified form of Dijkstra's shortest path algorithm. Loop closure in Atlas is accomplished by the addition of a link between local frames of reference with a transformation which can be estimated through correspondences between features in common between these two map frames. Atlas is a technique for *approximating* the true posterior map while maintaining near constant-time performance. Local maps are formed when the capacity of the previous map is filled with a pre-defined maximum number of features. This capacity based partition choice forms a cut which could potentially remove more information than a more informed choice. A partitioning scheme which minimizes the information removed from the map will generate a more accurate map.

2.2.1.4 Normalized Graph Cuts for Image Segmentation

Normalized graph cuts was used by Shi and Malik to provide better results than standard graph cuts on image segmentation tasks in the presence of outliers in Shi and Malik [2000]. Minimizing the cost of the graph cut often consists of a highly unbalanced partition where very few features will appear in one of the sets, and the

bulk will remain in the other. For making sub-maps, this unbalanced partition is undesirable because it will not allow the robot to remove many covariant landmarks, and the algorithm will not be able to handle large scale regions since most of the features will remain in the map which is being used. Instead of minimizing the cost of the cut, Shi and Malik developed a *normalized cut* which minimizes the disassociation measure shown in equation 3.

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \tag{3}$$

In equation 3, $cut(A,B)$ is the sum of all the weights of the edges removed by this cut, and $assoc(A,V)$ is the sum of all the weights of all edges connecting the set A to all of the vertices in the graph. Finding a partition $(A,B), A \bigcup B = V$ which minimizes this $Ncut(A,B)$ is the goal. Unfortunately, finding this partition is NP-hard; however, by relaxing the requirement that set membership is absolute, and allowing a vertex to be partly in one set and also partly in another, this minimization can be approximated by an eigenvalue problem. For completeness, please refer to the source paper Shi and Malik [2000] for the details of this computation and proof.

In terms of robot map building, normalized graph cuts will be used to divide the state vector into subsets which extract the typical compact block-diagonal structure of the covariance matrix.

2.2.2 Implementation

For the implementation we have adopted the SceneLib package provided by Davison [Davison and Murray, 2002, Davison, 2003, 1998, Davison et al., 2007]. This package allows for rapid prototyping and it provides a reference for evaluation of performance. The software used in this section has applied many of the advancements made to the open-source SceneLib package which have been detailed in the literature since its release in 2006. These advancements include the inverse depth parameterization from Montiel et al. [2006], and the sub-mapping technique Atlas from Bosse et al. [2003].

2.2.2.1 Monocular SLAM implementation

The intended application for SceneLib is that of performing SLAM on a monocular hand-held camera. With the addition of an odometric motion model, SceneLib can be made to work more accurately. This enables the visual SLAM algorithm to keep the small search regions for features even during larger scale loop closure events, which keeps the frame rate high.

Monocular SLAM typically requires a two-phase feature initialization process. A single observation of a feature only describes a ray along which this feature lies, it does not offer *any* information as to what the depth might be. This feature is initially represented as a line in MonoSLAM with a particle distribution over its length. Further measurements of this feature allow the particle set to be resampled at more likely positions until this set can be well represented with a Gaussian distribution – then the feature is fully initialized with this Gaussian as its representation. The problem with this, aside from the complexity of maintaining a two-phase feature initialization, is that if the feature is very far away it will not initialize well and cannot be represented as a Gaussian (since this Gaussian must include infinite depth with nonzero probability). Inverse depth behaves much better under the Gaussian distribution, shown in Montiel et al. [2006]. The two-phase initialization is not needed since now the feature can be initialized from one observation with a large depth uncertainty – one which includes infinite depth with significant probability. The filter will be able to use this feature for correcting heading errors when it is first observed, and once the feature converges to a better estimate it will also help correct spatial pose errors as well.

Despite the good performance of SceneLib in a small environment, the algorithm suffers from quadratic update complexity and will ultimately fail to maintain real-time operation when presented with a larger environment. This issue is addressed in the large-scale MonoSLAM implementation of Clemente et al. [2007] which details

a system very similar to the algorithm in this section. Their approach makes use of sub-maps by making a new local map when the current map reaches a capacity limit. The current feature measurements are added also to the new map and are maintained separately as independent features. These common features serve as a set of constraints which can be used to optimize the relative position of the sub-maps when closing loops. In our implementation, we have chosen an approach that is inspired by the Atlas [Bosse et al., 2003] sub-mapping framework.

2.2.2.2 Atlas implementation

Atlas leaves the choice of the loop-closure strategy open to the user. The implementation in Bosse et al. [2003] finds landmark correspondences and finds a robust transformation between their local frames of reference using RANSAC. The robot used in Bosse et al. [2003] is confined to a 2D plane and it uses a laser scanner to collect landmark measurements. Unfortunately, a search for corresponding landmarks does not extend well to the visual SLAM world of 3D features and images. Visual features are unlikely to be shared between two local maps. To adapt the Atlas scheme to the visual SLAM domain, we have employed the technique of projecting the features from the reference frame with which the loop closure is to take place into our current camera pose. This is accomplished by computing the pose of the robot in the other reference frame using the minimum uncertainty projection from Atlas and also computing the uncertainty in the pose using the rule given by $J_1(T_{ab}^{-1}, x_v^b)$, which is the Jacobian with respect to the first parameter of the composition transformation between the two map frames and the robot's current pose x_v^b in frame b, J_2 is likewise defined as the Jacobian with respect to the second parameter. The covariance of the composite transformation can be computed using formula 4. More details on the Jacobians J_1 and J_2 can be found in Tardos et al. [2002]. These are the same rules which are used in Atlas to compute the uncertainty along a given set of transformations.

$$\Sigma_{ac} = \begin{array}{c} J_1(T_{ab}^{-1}, x_v^b)\Sigma_{ab}J_1(T_{ab}^{-1}, x_v^b)^T + \\ J_2(T_{ab}^{-1}, x_v^b)\Sigma_{bc}J_2(T_{ab}^{-1}, x_v^b)^T \end{array} \tag{4}$$

The current Atlas strategy for splitting maps is to build a map until a finite capacity is reached. At this point, a sub-map is created. This strategy is essentially equivalent to making arbitrary cuts in the information shared between features until a partition is formed. A better strategy is to partition the set of features into smaller maps based on a cut which minimizes the information lost. This strategy will make a final *a posteriori* map which is closer to the ideal unpartitioned map than the *a posteriori* map from capacity Atlas.

Adapting the normalized graph cuts [Shi and Malik, 2000] algorithm from the image segmentation community to be used for partitioning robot map features is accomplished by determining that the affinity measure between features is the volume of the information which connects them. This is a simple computation from the covariance matrix. The only terms of interest are the $I_{y_i y_j}$ terms which measure the information shared between two features y_i and y_j. The weight on the edge connecting feature y_i to feature y_j is set to $W_{ij} = \left|\det I_{y_i y_j}\right|$ which measures the total information shared between these two features, regardless of their parameterization. Now the normalized graph cuts algorithm can partition these two sets into balanced sets which have the minimum information shared along the cut. From Shi and Malik [2000], we set D to be the diagonal matrix with $D_{ii} = \sum_j W_{ij}$ and then we can solve the eigenvalue problem:

$$(D - W)y = \lambda D y \tag{5}$$

to find the real-valued eigenvector y corresponding to the second-smallest eigenvalue γ. Thresholding the elements of y gives a partitioning on the set of features: if $y[i] > 0$ then feature y_i is in set 1, and if $y[i] < 0$ then feature y_i is in set 0.

2.2.2.3 Map Partitioning

The transformation between the old map and the new map is initially set to the old robot pose and covariance in the old map, just as with capacity based Atlas. The robot pose in the new frame is set to zero, with zero covariance Σ_{xx}.

Unlike in capacity based Atlas, partitioning based on normalized graph cuts requires actually splitting up an active map to generate two sub-maps rather than simply starting a new map. Fortunately, this just involves re-expressing the features which will be carried into the new map frame in its coordinate system, and transforming the covariance matrix by this transformation.

Features y_i which are to be moved from frame a to b are transformed according to the rule $y_i^b = T_{ab} y_i^a$ Since y_i are expressed in a spherical "inverse depth" coordinates, this transformation involves re-expressing the origin in terms of the new coordinate system b and rotating the horizontal bearing θ and vertical bearing ϕ into the new representation. The inverse depth coordinate ρ remains unchanged.

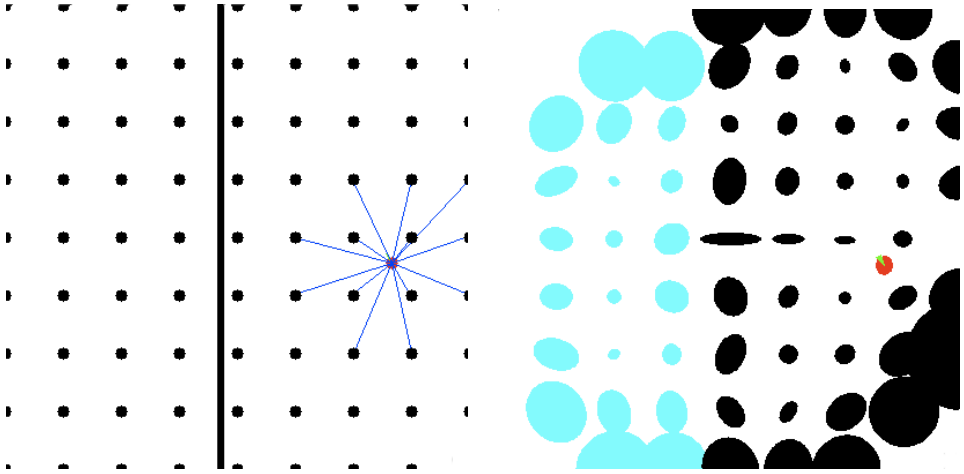
Feature covariance elements $\Sigma_{y_i y_j}$ are set to zero and are removed if y_i and y_j fall into separate partitions. $\Sigma_{x y_i}$ is transformed according to the rule seen in equation 6

$$\Sigma_{x y_i}^b = \frac{\delta x^b}{\delta x^a} \Sigma_{x y_i}^a \frac{\delta y_i^b}{\delta y_i^a}^T \tag{6}$$

The only type of covariance submatrix which remains is $\Sigma_{y_i y_j}$ where y_i and y_j fall within the same partition (or are the same element if $i == j$) in which case the transformation rule is seen in equation 7.

$$\Sigma_{y_i y_j}^b = \frac{\delta y_i^b}{\delta y_i^a} \Sigma_{y_i y_j}^a \frac{\delta y_j^b}{\delta y_j^a}^T \tag{7}$$

Using these transformation rules, the new covariance and state vectors can be constructed for the partitioned sub-map. The robot is now able to proceed with its task using this new local map.



(a) Set up for simulated experiment.

(b) Result of normalized graph cuts on proof-of-concept. The blob colors correspond to which sub-map each one is in. The size of each blob corresponds to the associated uncertainty in its location.

Figure 1: Simulated experiment demonstrating partition determined by normalized graph cuts

2.2.3 Results

A proof-of-concept simulation was created to validate the use of the normalized graph cuts algorithm for the application of splitting up robot maps. This simulated world is a typical 2D planar robot which makes range and bearing measurements on a set of features, which are located on a regular grid. An artificial "wall" has been placed in the center of the world which splits the features on the right half from the features on the left half as can be seen in figure 1(a). This wall prevents the robot from measuring features from the opposite side, which approximates the effect of a wall in the real world separating two rooms, whose features can be independently measured and may be a good candidate for partitioning into sub-maps.

In this initial proof of concept experiment, the partitioning algorithm was able to immediately segment the features according to the most natural partition suggested by the wall. This result can be clearly seen to correspond exactly to the wall in figure 1(b).

The second experiment is to apply this algorithm to the MonoSLAM application with Atlas sub-mapping and see if the partitions also follow natural architectural boundaries. SceneLib does not come with a joint data association gating so it does make data association errors when used in the large scale problem, generating inconsistent maps. The branch-and-bound technique featured in Clemente et al. [2007] will be implemented in the immediate future but was not available at the time of writing. This will enable robust joint data association and make the software functional for robot mapping in large areas.

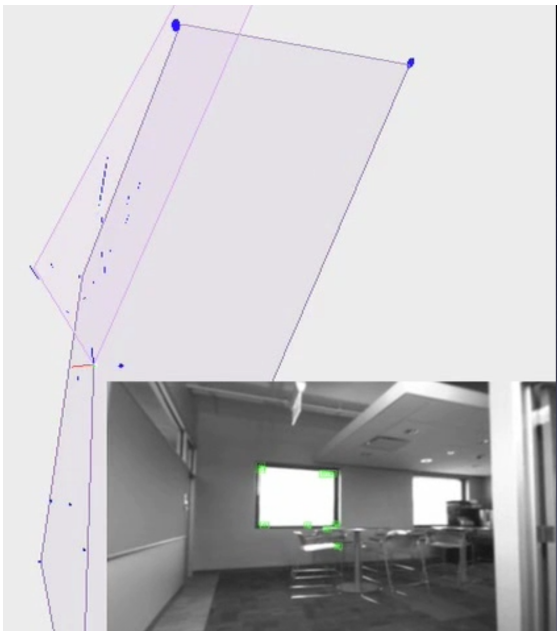
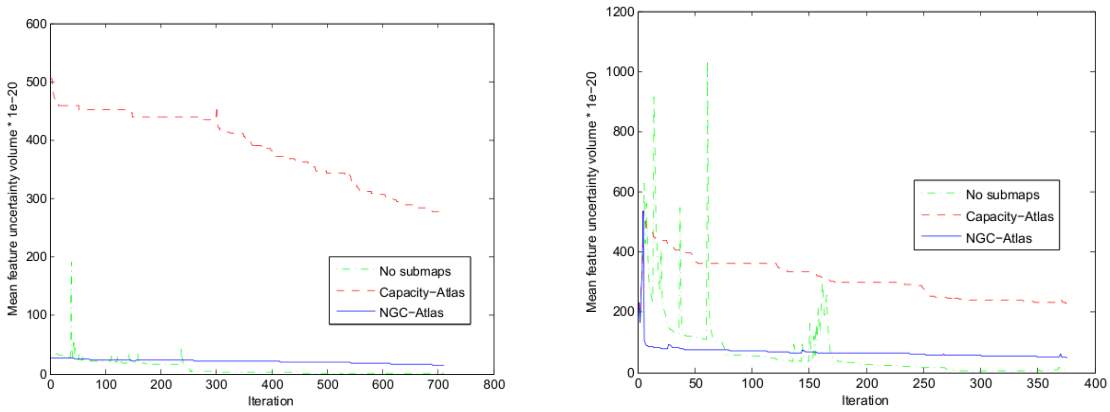


Figure 2: A top down view of the map. The normalized graph cuts algorithm finds a map partition which is related to the topology of the space. The new map partition contains mostly features from the new area that the robot is entering.

A small experiment was conducted where the robot moved from one room into the adjacent hallway. The normalized graph cuts algorithm detected that the map supported a graph cut which removed a small amount of information and performed this cut, effectively partitioning out the hallway. This result is shown in figure 2. Features are shown as green ellipsoids when their uncertainty is small enough and shown as red when the uncertainty is large. Purple ellipsoids signify that these features have

been partitioned into a new local map – the partition shown in the video supports the fact that the normalized graph cuts algorithm can find partitions which follow architecturally significant boundaries. Since the robot can see some of the features through the doorway, they have become correlated with other features from the first room. This causes some features to fall on the wrong side of the graph cut.



(a) Slow motion.

(b) Fast motion.

Figure 3: Mean feature uncertainty with various sub-mapping options, simulated landmarks.

The third experiment it to establish the advantage of using normalized graph cuts in place of capacity for making Atlas map partitions. Since the current software lacks joint data association, this result was established with a full-scale simulation with perfect data association in the 3D MonoSLAM framework. Since Atlas is effectively making arbitrary partitions, and it is potentially removing *stronger* information content edges while the normalized graph cuts algorithm will strongly prefer removing *weaker* information content edges, it should be expected that partitioning with normalized graph cuts will generate maps with less information loss than maps generated with capacity cuts algorithm in vanilla Atlas. Refer to figures 3(a) and 3 for a graphical comparison of the average feature uncertainty with the three sub-mapping options. These two experiments were collected with two separate trajectories with real robot odometry in simulated feature environments. This result is established by comparing

the uncertainty in the maps from runs with no sub-mapping, with Atlas, and with normalized graph cuts Atlas.

The normalized graph cuts implementation allows for significantly more certain feature localization compared to arbitrary capacity cuts. This result demonstrates that normalized graph cuts offers significant advantage over arbitrary cut choices as implemented in standard Atlas.

2.2.4 Conclusions

Traditional approaches to SLAM are limited by a quadratic complexity. To address this problem, sub-mapping approaches have been suggested. These methods typically apply a heuristic for the partition selection. In this section we have presented the use of normalized graph cuts as a methodology for the partitioning of sub-maps. The application of normalized graph cuts for map partitioning has been demonstrated in the context of monocular SLAM using an Atlas type strategy. The proposed method has been implemented in an extension to SceneLib. Preliminary experiments using both simulation and real data clearly indicate that this method generates maps that are intuitively a correct partition. Consequently, the method provides a theoretically sound basis for SLAM using sub-maps and as such, directly addresses the complexity issue while minimizing the information omitted from SLAM.

2.3 SLAM with EM for moveable object tracking[3]

Many SLAM algorithms rely on the assumption that the environment is static, and will perform poorly or fail if mapped landmarks move. However, to operate in real world dynamic environments, algorithms will need to recognize moveable objects. Consider, for example, a robot that makes a map of a room, and then returns several days later. Some of the features in its map might correspond to immobile objects,

[3] ***This section is based upon Rogers III et al. [2010]***

such as walls, while some might correspond to objects that may have moved, such as furniture. If someone has moved some of the objects in the room that were part of the robot's map, it will be potentially catastrophic because the SLAM system will make an inconsistent map out of incompatible measurements.

In this section, we propose an EM approach to data association and static vs. moveable object determination for performing SLAM in a pathological office environment where mapped landmarks move. Our algorithm has been validated in an indoor environment which is mapped by a mobile robot.

A common assumption is that the environment being mapped is static. There are two main research directions which attempt to relax this assumption. One approach partitions the model into two maps; one map holds only the static landmarks and the other holds the dynamic landmarks. Hähnel *et. al.* use an Expectation Maximization (EM) based technique to split the occupancy grid map into static and dynamic maps over multiple iterations in a batch process. This technique is shown in Hähnel et al. [2002] and Hähnel et al. [2003] to be effective in generating useful maps in environments with moving people. Biswas *et. al.* take a finite set of snapshots of the map and employ an EM algorithm to separate the moving components to generate a map of the static environment and a series of separate maps of the dynamic objects at each snapshot. Wolf and Sukhatme [2005, 2004, 2003] are able to separate static and dynamic maps with an online algorithm. Stachniss and Burgard developed an algorithm [Stachniss and Burgard, 2005] which identifies dynamic parts of the environment which engages a finite number of states. The map is represented with a "patch map" that identifies the alternative appearances of the portions of the map which are dynamic, like doors.

The second direction with respect to relaxing the static world assumption is to track moving objects while mapping the static landmarks. Wang and Thorpe [2002], Wang et al. [2003] are able to track moving objects and separate the maps in an online

fashion by deferring the classification between static and dynamic objects until several laser scans can be analyzed to make this determination.

Bibby and Reid [2007] use an EM based technique over a finite time-window to perform dynamic vs. static landmark determination; however, their approach differs from ours in several important ways. First, they use a finite sliding window after which no data associations can be changed. Our approach has no such limitation as we are attempting to determine which aspects of the environment are static vs dynamic over the entire mapping run. Bibby's technique does a good job of detecting *moving* objects but it will not be able to detect *moveable* objects over longer time intervals, that might move when they aren't being observed by the robot. Additionally, Bibby addresses the static data association problem by maintaining a distribution of data associations across the sliding window; the data association decision is made permanent at the end of the window (6 steps in Bibby's implementation). An infinite sliding window would be computationally intractable in this implementation due to exponential growth of the interpretation tree; however, our approach offers an alternative solution to the static data association problem which does not suffer from finite history.

2.3.1 Approach

Our algorithm is based upon the Square Root SAM of Dellaert [Dellaert, 2005]. We have modified this algorithm with a per-landmark weighting term which enables discrimination between stationary and mobile landmarks to allow for more reliable localization. The remaining landmarks which have a low weight are classified as being moveable and are now tracked by the robot without influencing the robot's trajectory.

2.3.1.1 Square Root SAM

Our implementation of Square Root SAM finds the assignment for the robot trajectory and landmark positions that minimizes the least squares error in the observed

measurements. As is common in the SLAM literature, our motion and measurement models assume Gaussian noise. Each adjacent pose in the robot trajectory is modeled by the motion model in equation 8.

$$x_i = f_i(x_{i-1}, u_i) + \nu_i \tag{8}$$

where $f_i(.)$ is the nonlinear motion model and u_i is the observed odometry from the robot, and ν_i is the process noise. In our case, we use a differential drive robot which has a three-dimensional pose (x_i, y_i, θ_i). The model is shown in equation 9.

$$\begin{pmatrix} \Delta x_i \\ \Delta y_i \\ \Delta \theta_i \end{pmatrix} = \begin{pmatrix} u_0 \cos(\tilde{\theta}) - u_1 \sin(\tilde{\theta}) \\ u_0 \sin(\tilde{\theta}) + u_1 \cos(\tilde{\theta}) \\ u_2 \end{pmatrix} \tag{9}$$

where u_0 is the forward motion, u_1 is the sideways motion, u_2 is the angular motion of the robot, and $\tilde{\theta} = \theta_{i-1} + \frac{u_2}{2}$. The measurement model determines the range and bearing to the landmarks. It has the form shown in equation 10.

$$h(x_i, l_j) = \begin{pmatrix} \sqrt{(x_i - l_j^x)^2 + (y_i - l_j^y)^2} \\ \tan^{-1}\left(\frac{(l_j^y - y_i)}{(l_j^x - x_i)}\right) - \theta_i \end{pmatrix} \tag{10}$$

The linearized least squares problem is formed from the Jacobians of these motion and measurement models as is seen in Dellaert [2005]. By organizing the Jacobians appropriately in matrix A and collecting the innovation of the measurements and odometry in vector b we can iteratively solve for the robot trajectory and landmarks which are stacked in Θ as seen in equation 11.

$$\Theta^* = \arg\min_{\Theta} \|A\Theta - b\|^2 \tag{11}$$

After each iteration, the Jacobians are re-linearized about the current solution Θ. The solution to this minimization problem can be found quickly by direct QR factorization of the matrix A using Householder reflectors followed by back-substitution.

The source paper for this technique Dellaert [2005] exploits sparsity to vastly improve performance; however, we are currently using dense matrices. The optimization currently runs in approximately one second per iteration for a SAM problem of around 50 poses and 100 measurements with dense matrices.

2.3.1.2 Expectation Maximization

To establish an EM algorithm for SAM with moveable objects, we first must express the joint probability model in equation 12.

$$P(X, M, Z) =$$

$$\prod_{poses} P(x_i|x_{i-1}, u_i) * \prod_{landmarks} P(z_k|x_{i_k}, l_{j_k}) \tag{12}$$

where $P(x_i|x_{i-1}, u_i)$ is the motion model and $P(z_k|x_{i_k}, l_{j_k})$ is the sensor model. We add a hidden variable ω_k to each landmark measurement, which changes the joint probability model to equation 13.

$$P(X, M, Z, \Omega) =$$

$$\prod_{poses} P(x_i|x_{i-1}, u_i) * \prod_{landmarks} P(z_k|x_{i_k}, l_{j_k}, \omega_k) \tag{13}$$

With a Gaussian representation for the sensor model, this new set of parameters ω_k results in the sensor model in equation 14.

$$P(z_k|x_{i_k}, l_{j_k}, \omega_k) \propto$$

$$\exp -(\omega_k((z_k - h(x_{i_k}, l_{j_k})^T \Sigma^{-1}(z_k - h(x_{i_k}, l_{j_k}))) \tag{14}$$

With the interpretation that ω_k is the likelihood that this measurement comes from a static landmark, if the landmark is not static (i.e. $\omega_k = 0$), then the measurement does not affect the joint likelihood since $P(z_k|x_{i_k}, l_{j_k}, \omega_k) = 1$ for all assignments to the robot poses and landmark positions. When the landmark is static (i.e. $\omega_k = 1$), then this weighting term makes this measurement behave like normal. Obviously,

the hidden variables ω_k cannot be directly observed by the robot and must instead be estimated from multiple observations of each object. The M step selects new assignments for the ω_k to maximize the joint likelihood. Since the likelihood can be trivially maximized by setting all $\omega_k = 0$, we introduce a Lagrange multiplier to penalize setting too many moveable landmarks. The non-constant portions of the log likelihood for the measurements is now seen in equation 15.

$l(Z, X, L, \Omega) =$

$$\sum_{measurements} (-\omega_k(\eta_k^T \Sigma_k^{-1} \eta_k)) - \lambda(1-\omega)^T(1-\omega) \tag{15}$$

where η_k is the innovation of the k-th measurement (the k-th measurement minus its predicted value). This log likelihood is maximized when

$$\frac{\delta l(Z, X, L, \Omega)}{\delta \Omega} = 0 \tag{16}$$

For each ω_k we get the equation 17

$$-(\eta_k^T \Sigma_k^{-1} \eta_k) + 2\lambda - 2\lambda\omega_k = 0 \tag{17}$$

so

$$\omega_k = 1 - \frac{\eta_k^T \Sigma_k^{-1} \eta_k}{2\lambda} \tag{18}$$

We have made the additional modification that the ω_k is not assigned per measurement, but instead since these measurements come from objects we would like to treat the objects as the things that are moveable instead of the measurements being unreliable. This is a simple modification which changes equation 15 into equation 19.

$l(Z, X, L, \Omega) =$

$$\sum_{measurements} (-\omega_{l_k}(\eta_k^T \Sigma_k^{-1} \eta_k)) - \lambda(1-\omega)^T(1-\omega) \tag{19}$$

where ω_{l_k} is the weight of landmark l involved in the kth measurement. For each ω_{l_k} we get the equation

$$\sum_{k \in K_l} -(\eta_k^T \Sigma_k^{-1} \eta_k) + 2\lambda - 2\lambda\omega_l = 0 \tag{20}$$

so

$$\omega_l = 1 - \frac{\sum_{k \in K_l} \eta_k^T \Sigma_k^{-1} \eta_k}{2\lambda} \tag{21}$$

where K_l is the set of measurements of landmark l. The Lagrange multiplier λ can be assigned to trade off the penalty for having moveable landmarks. This was the M step of EM.

The E step of EM computes the robot trajectory and the landmark positions with the current estimates of the weighting terms ω_l. The least-squares problem solved by Square Root SAM to find the most likely map is simply the weighted least squares problem, which is to add a weighting term $\omega_l \in [0, 1]$ to each landmark measurement row i.e.

$$H * x_i + J * l_{j_i} = z_{ij} - h(x_i, l_{j_i}) \tag{22}$$

becomes

$$\omega_{l_{j_i}} * (H * x_i + J * l_{j_i}) = \omega_{l_{j_i}} * (z_{ij} - h(x_i, l_{j_i})) \tag{23}$$

where $H = \frac{\delta\nu}{\delta x_i}$ and $J = \frac{\delta\nu}{\delta l_{j_i}}$ with $\omega_{l_{j_i}}$ is the weight assigned to the landmark which we are measuring in this row. In the least squares formulation, this will have the effect of scaling the contribution of this measurement to the overall solution.

2.3.1.3 Moveable Landmark Tracking

After the terminal iteration of the EM algorithm, landmarks which have a weight factor falling below a specific threshold are removed from the SAM optimization and collected in a separate data structure. The final map is optimized once again with the moveable landmarks removed. Measurements on the dynamic landmarks are used with the final trajectory to compute global locations for the dynamic landmarks. These landmarks are now moved into a separate list of moveable landmarks where

they could be referred to later to find a list of observed positions. If the moveable landmarks were tagged with semantic information, the robot would then be able to use this data structure as a candidate list of search locations for the object for a retrieval task. The robot can start with the most recently seen position for this object and then try the other places that the object has been seen in the past. Currently, the distribution of positions of the moveable landmarks is being represented as a list of observed locations.

2.3.2 Experiments

To verify the performance of our algorithm, we performed a series of experiments with moveable landmarks in our office environment.

2.3.2.1 Robot Platform

To test our system, we collected data with a Mobile Robotics Peoplebot. Our robot is equipped with a SICK LMS-291 laser scanner, as well as a Logitech webcam. As measurements, we detect ARToolKit Plus [Wagner and Schmalstieg, 2007] markers in the camera images. The ARToolKit was used in these experiments because the emphasis here is on the detection of static and mobile landmarks. Having landmarks with trivial data association helps us focus on the key contribution of this section; however, there is no loss of generality and natural landmarks will be considered in future work. The resulting measurements give the relative pose of each marker with respect to the camera. Laser data was also logged, but is only used for visualization purposes. Wheel odometry is also logged, and is used as the input for the motion model.

2.3.2.2 Experimental Setup

The environment used for our experiment was a portion of our lab, consisting of two student offices and the corridors connecting them. ARToolKit markers were

Figure 4: The robot platform used for these experiments. The webcam attached to the laptop is the one that is used to collect visual measurements.

placed throughout the environment to serve as landmarks. Some of the markers were pinned to the walls, while others were held by moveable frames which facilitated their movement during the experiments.

2.3.2.3 Procedure

Our first experiments were to move the robot in a circle in one of our student offices measuring 4.5 meters on a side. This office had 12 ARToolKit markers pinned to the walls. In addition to these static landmarks, this office had a total of 10 moveable landmarks which were placed upon the desks and shelves. We performed tests of our implementation of the standard SAM algorithm by moving the robot in this office to collect measurements of landmarks without moving them during the test run. The next experiment was to move the landmarks to a second location within this same cubicle midway through the data collection. We performed a larger scale experiment in which the robot moved between both of our group's student offices. Each of these offices is 4.5 meters on a side, and they are separated by about 12 meters of corridors. We left the 12 ARToolKit markers in the first office from the small scale experiment, and placed 9 markers in the second office. Additionally, we pinned 8 markers to the

walls in the corridor between our offices. Several data sets were collected in this setup with varying numbers of moveable landmarks. In each run, the robot was moved in the first office so that each landmark was observed multiple times and then the robot was driven down the corridor. While the robot was being moved down the corridor, the moveable landmarks were transferred from the starting office to random locations in the second office. The robot was then maneuvered in the second office so that each landmark was observed multiple times and then it was driven back to the first office. The robot was driven in the first office in a few loops and was finally placed as close as possible to its starting location. Each test run of this type featured similar trajectories, but always the moveable landmarks were placed in arbitrary positions in the two offices.

The logs were used as input for our algorithm. While the ARToolKit Plus measurements provide the relative pose of the landmark in Cartesian space with respect to the camera, we instead converted this to a range and bearing measurement. As described in section 2.3.1, the algorithm first considered all measurements, and iteratively adjusted the weights to determine which landmarks were moveable, and which were static. This iteration was repeated until convergence to a specific threshold. Once a stable configuration had been found, the landmarks which had weights below a certain threshold were removed from the SAM problem and were tracked separately. At this point, a final map was generated.

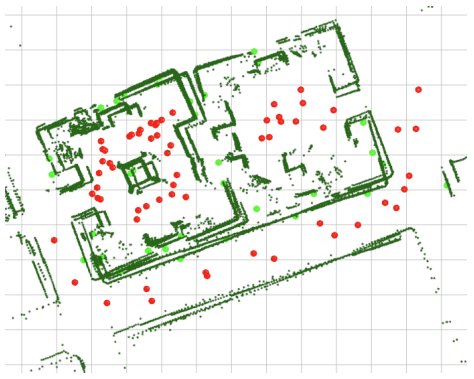
2.3.3 Results

We present the longest test run in detail. The initial state of the problem can be seen in Figure 6(a). This corresponds to the raw odometry and sensor measurements before the map is optimized. The robot starts out in the upper rightmost corner of the left office, facing up in the image. The initial iteration of the EM algorithm will have 1.0 in each ω_k, so each landmark is initially assumed to be static. The

Figure 5: One of the images used as part of our experiments, as taken from the robot. The AR markers on the wall are static, while the ones on the desk and refrigerator can be moved

SAM optimization is iterated until convergence with these parameters, resulting in a very poor quality map as can be seen in Figure 6(b). It is apparent in this figure that the moveable landmarks have resulted in incorrect loop closures causing the two offices to be pulled on top of each other. After two iterations of the EM algorithm the map can be seen in Figure 6(c). This map is clearly better than the initial state; with two additional iterations of the EM algorithm the low weight landmarks can be thresholded to generate the final map in Figure 6(d). In this particular run, there were 10 moveable landmarks, 6 of which were detected as moveable. No static landmarks were mistakenly detected as moveable. The remaining 4 moveable landmarks which were mistakenly classified as static were only observed in one of the two offices. Without the observation of the landmark in its second position, the algorithm cannot determine that the landmark had moved. The missing observations can be explained by our use of a webcam and some poor lighting, or the missing landmarks do not appear with a front aspect view which ARToolKit Plus can detect. We have performed two additional test runs of this length and four runs of the single office test, with similar results.

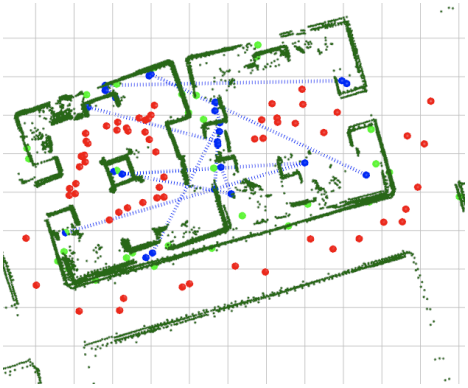
We performed an additional test run where we ignore measurements from the static landmarks. This test was generated by running one of our normal test runs



(a) The initial state of the map, before optimization.

(b) The resulting map with all measurements (including moveable objects) prior to the first weight assignment.

(c) The resulting map after two iterations of the EM algorithm.



(d) The resulting map after all iterations and thresholding to exclude moveable objects.

Figure 6: Poses are shown in red, static landmarks are shown in green, and moveable landmarks are shown in blue, connected by a blue dotted line showing their movement. The dark green points are laser scans, and are for visualization purposes only.

with measurements suppressed from markers that we know to have been static. In this test run, the EM operation was able to correctly identify all of the landmarks as moveable. The final output appears the same as the initial odometry solution. This makes sense because the SAM problem has no measurements between landmarks which affect the trajectory since all of the landmarks had moved.

2.3.4 Conclusions

While our algorithm worked well for the scenarios we tested, there are some cases that could be more problematic for this technique. Here we provide a brief discussion of some scenarios in which this technique might not perform well.

One case that could cause difficulties for this technique is if several landmarks moved together in a coherent manner. For example, if many landmarks were to move one meter in the same direction, the algorithm might not factor these out, as it might be more likely that the error could be ascribed to poor odometry. This case is of particular interest, because this is what would happen if we were to track many features that were part of a single object. As the object moved, all of the features would undergo the same rigid body transform, moving them in a coherent manner. A possible solution to this would be to track the entire object, instead of individual features on the object.

Because we determine which objects moved based on their residuals, if a landmark were to move only by a small amount, the residual would probably not be large enough to cause it to be excluded. In this case, it would be used for our algorithm, and would add error to the final map and trajectory.

Another potentially troubling scenario is if most or all of the landmarks we detect are moveable. In our experiment, temporary visual features were used as landmarks for the robot. In practice, some more permanent architectural landmarks should be chosen in addition to potentially moveable landmarks, such as walls.

Also, data association was not an issue because ARToolKit markers were used, and so different markers that appeared near each other were never considered as the same landmark. If we were to use natural features as landmarks, the moveable landmark detection problem becomes much more complex. However, if most of the landmarks we see are static and only a few have moved, a similar technique should still be applicable.

We have presented an algorithm which allows for the relaxation of the static world assumption in SLAM. We provided experimental results based upon real robot data and measurements of artificial landmarks. These results serve as a first step towards moveable landmark detection and tracking with real feature measurements and data association. This algorithm should help mitigate the effect of data association errors without the limitation of finite windows for reversible data association.

2.4 OmniMapper

The study presented in section 2.3 was based upon an implementation of Square-Root Smoothing and Mapping $\sqrt{SAM}$ from Dellaert [2005]. The software that was written for this experiment was implemented as a proof-of-concept; it was not optimized to leverage the sparse structure of $\sqrt{SAM}$. We replaced our implementation of $\sqrt{SAM}$ with the *GTsam* library developed at Georgia Tech to gain these and other advantages in optimizing maps via sparse operations in $\sqrt{SAM}$. This new software suite is a library for mobile robot mapping called *OmniMapper*. We have used it to develop virtual measurements [Trevor et al., 2009], learned object recognition mapping [Rogers III et al., 2011], for multi-robot mapping [Rogers et al., 2011], and to determine mapping performance with sensory degradation [Rogers et al., 2010]. In addition, *OmniMapper* is used by the Army Research Laboratory's man-portable robotics research group; an open source release is also planned.

OmniMapper uses the *GTsam* library to optimize a graph of measurements between robot poses along a trajectory, and between robot poses and various landmarks in the environment. Measurements come from various software components. Measurements of simple objects like points, lines, and planes are data associated to mapped landmarks with the joint compatibility branch and bound (JCBB) technique from Neira and Tardós [2001]. Measurements of richer landmarks such as

objects or signs are data associated based upon interpretation of this semantic information [Rogers III et al., 2011].

OmniMapper is a complete SLAM solution performing feature-based as well as pose graph SLAM, complete with probabilistic data association and loop closure. A complete SLAM solution consists of a *back-end* as well as a *front-end*; these two components must work together to build a map of an environment. There are many options for back-end SLAM libraries; however, they can be described by the two basic categories of *Filtering* approaches like the EKF and *Smoothing* approaches like $\sqrt{SAM}$. These options were described in section 2.1

OmniMapper uses the *M-space* representation from Folkesson et al. [2007] in a plugin architecture for each type of landmark measurement to be used. The *M-space* representation makes incorporating new landmark measurement types simple by expressing linearization parameters in terms of simpler components via the chain rule and then multiplying the matrices together to form the linearized measurement model Jacobians. We have implemented plugins for point features, lines, planes, objects, 2D canonical scan matching (CSM) [Censi, 2008], and full point cloud alignment via generalized iterated closest point (G-ICP) [Segal et al., 2009].

Since *OmniMapper* is a complete SLAM solution, it also contains SLAM *front-end* tools for data association as well as integrating proprioception to set favorable initial conditions for map optimization. In addition to these components, *OmniMapper* also contains tools which help make the SLAM process useful to a mobile robot in live operation. *OmniMapper* provides a correction to the odometric pose estimate to keep track of the robot's pose in a fixed map frame. This enables long-range navigation in autonomous operation. In addition, the *OmniMapper* can provide an occupancy grid map or 3D point cloud of its environment depending on the sensors used to measure features.

2.4.1 Feature based mapping

The next several sections will describe the plugins which have been developed for feature-based mapping. Feature-based mapping, as opposed to featureless mapping, builds both a map of the robot's trajectory as well as a description of the landmarks in the map.

2.4.2 Line mapping (2D walls)

We use the RANSAC [Fischler and Bolles, 1981] based technique in Nguyen et al. [2005] to extract 2D lines in laser data which represent walls. The first step of this technique is to back-project the laser range readings to 2D points. For fifty iterations, the algorithm proceeds as follows. First, two points are selected uniformly from the remaining laser points. A consensus set is extracted from the remaining points by checking that the point-line distance is below a threshold of 5 cm. These points are tested to form a line by making sure that there aren't any gaps larger than 0.8 meters. This gap is set to be large so that the lines are able to jump across doorways to form long lines. If the points which lie on this line form a line segment of sufficient length (1.5 meters), then the points in consensus with this line segment are removed from future consideration and this line segment is provided as a measurement to the mapper. A visualization of the extracted line is shown in figure 7.

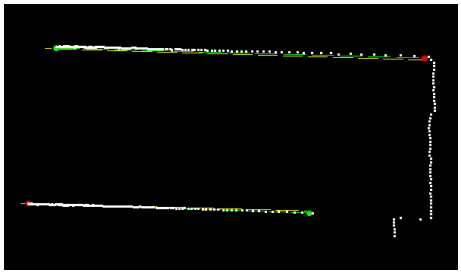


Figure 7: Laser scanner data from the robot. The structure of the hallway is recognized by the laser line extractor as two walls on either side, shown with green lines. The wall at the end of the hall is too small in this view to be recognized as a line. Raw laser points are shown in white.

The M-space feature representation is useful for expressing measurements of partially observed objects, such as walls seen with limited range or with environmental occlusion. The M-space representation effectively corrects based upon the perpendicular distance and relative angle between the robot pose and the wall in this implementation. It is possible to upgrade this representation to include the endpoints as they are fully observed; however, this is reserved for future work. Currently, the measurement correction corresponds to the angle and range to the wall, $\eta = (\phi, \rho)$. Endpoints, along with angle and range, are used for data association. The factor graph representation requires the Jacobians of these measurements which are $\frac{\delta\eta}{\delta x_r}$ and $\frac{\delta\eta}{\delta x_f}$ where x_r is the robot pose and x_f is the global landmark pose. The M-space feature representation uses the chain rule to represent Jacobians in terms of smaller building blocks which are re-usable between different types of features.

In the M-space feature representation, the Jacobians are broken down in terms of local parameterizations of the landmarks (x_o) to $\frac{\delta\eta}{\delta x_r} = \frac{\delta\eta}{\delta x_o}\frac{\delta x_o}{\delta x_r}$. The transformation from the local reference frame x_o to the global frame x_f is a simple rigid body transformation. The Jacobian of this transformation is the derivative of the local representation x_o with respect to the robot pose x_r which is $\frac{\delta x_o}{\delta x_r} = \begin{bmatrix} -1 & 0 & x_{o_y} \\ 0 & -1 & -x_{o_x} \end{bmatrix}$

For completeness, the Jacobian $J_{\eta o}$ for walls is provided here. Let $\left(x^1_{o_x}, x^1_{o_y}, x^2_{o_x}, x^2_{o_y}\right)$ be the coordinates of the endpoints of the walls. Let dy be the y-coordinate difference in the endpoints of the wall $x^2_{o_y} - x^1_{o_y}$, and let dx be the x coordinate difference $x^2_{o_x} - x^1_{o_x}$. Also, let L be the length of the wall, $L = \sqrt{dx^2 + dy^2}$.

$$J^T_{\eta o} = \frac{\delta\eta^T}{\delta x_o} = \begin{bmatrix} \frac{-dy}{L^2} & \frac{-dy*\left(dx*x^2_{o_x}+dy*x^2_{o_y}\right)}{L^3} \\ \frac{dx}{L^2} & \frac{dx*\left(dx*x^2_{o_x}+dy*x^2_{o_y}\right)}{L^3} \\ \frac{dy}{L^2} & \frac{dy*\left(dx*x^1_{o_x}+dy*x^1_{o_y}\right)}{L^3} \\ \frac{-dx}{L^2} & \frac{-dx*\left(dx*x^1_{o_x}+dy*x^1_{o_y}\right)}{L^3} \end{bmatrix} \tag{24}$$

Equation 24 is the Jacobian relating the error from the robot relative position with

respect to the landmark.

$$\frac{\delta\eta}{\delta x_f} = \frac{\delta\eta}{\delta x_o}\frac{\delta x_o}{\delta x_f}\tilde{B} \tag{25}$$

Equation 25 shows the overall Jacobian relating error from the global representation with respect to the landmark. In this equation, $\tilde{B}$ is the binding matrix which projects corrections from the M-space representation (ϕ, ρ) to the endpoint representation x_f. In this way, the corrections will only be applied to the direction and distance of the line, but not to the endpoints themselves. The M-space paper [Folkesson et al., 2007] should be consulted for more implementation details.

2.4.3 Door mapping

[4] For door (and windows) feature detection a combination of laser range data and intensity data from the images is used. Initially images are processing using a Canny edge detector [Canny, 1986]. A histogram of vertical edges is generated. Long vertical edges are identified from the histogram. The laser and image data have been pre-calibrated. Using hypothesized door posts, it is possible to check for the presence of depth discontinuities in the laser range scan. At the same time the distance between door posts can verified from the laser scan. The door post are required to at a certain separation to accept a door hypothesis. The position of the door is specified by the location of the center of the door with respect to robot frame of reference. A normal for the door is defined to point into the hallway, to enable identification of which side of a door that has been seen. An example of a detected door is shown in Figure 8. Additional details as the underlying principles for object detection are available from Ma [2010].

[4] The door mapping component was developed with Jeremy Ma at JPL.

Figure 8: Example of door detection in a hallway setting

2.4.4 Plane mapping

Plane mapping is essentially an extension of line mapping, described in section 2.4.2, into three dimensions. As in the line mapping case, an error function is described between a robot pose x_r, and a landmark representation. Plane landmarks are represented with a surface normal and a distance to the origin, $(\vec{n}, d)$, and measurements are given by $(\vec{n}_m, d_m)$. The measurement error function to be minimized via nonlinear optimization in the GTsam framework is given by:

$$h = \begin{pmatrix} R^T * \vec{n} \\ \langle \vec{n}, t \rangle + d \end{pmatrix} - \begin{pmatrix} \vec{n}_m \\ d_m \end{pmatrix}$$

The Jacobian with respect to the robot pose is then given by:

$$\frac{\delta h}{\delta X_r} = \begin{bmatrix} 0 & -n_a & n_b & \\ n_a & 0 & -n_c & [0] \\ -n_b & n_c & 0 & \\ 0 & 0 & 0 & \vec{n}^T \end{bmatrix}$$

The Jacobian with respect to the landmark is given by:

$$\frac{\delta h}{\delta n_{map}} = \begin{bmatrix} [R_r] & \vec{0} \\ \vec{X}_r^T & 1 \end{bmatrix}$$

2.4.5 Plane mapping in 2D and 3D

In Trevor et al. [2012], we developed a version of the laser line mapping plugin which provides measurements on 3D planes. This allows this version of the laser line plugin to be used together with the plane plugin to extract additional measurements of planes in the environment. Line measurements on planes provide only two constraints on a mapped plane feature: a range constraint and an angular constraint. The angular constraint is that the normal to the map plane forms a right angle with a vector along the measured line. Given a robot pose X_r, a transform from the map frame to the robot frame in the form of $(R, \vec{t})$, a previously observed feature in the map frame $(\vec{n}, d)$ and a measured line with endpoints p_1 and p_2 where $\vec{b} = \frac{p_1 - p_2}{|p_1 - p_2|}$ is a unit vector along the measured line and $\bar{p} = \frac{p_1 + p_2}{2}$ is the midpoint of the line, the measurement function for laser lines h is given by:

$$h = \begin{pmatrix} \left\langle R^T * \vec{n}, \vec{b} \right\rangle \\ \langle \vec{n}, \bar{p} \rangle + d \end{pmatrix}$$

The Jacobian with respect to the robot pose is then given by:

$$\frac{\delta h}{\delta X_r} = \begin{bmatrix} \vec{p_1} & 0 \\ \bar{p} & 1 \end{bmatrix} * \begin{bmatrix} 0 & -n_a & n_b & \\ n_a & 0 & -n_c & [0] \\ -n_b & n_c & 0 & \\ 0 & 0 & 0 & \vec{n}^T \end{bmatrix}$$

The Jacobian with respect to the landmark is given by:

$$\frac{\delta h}{\delta n_{map}} = \begin{bmatrix} \vec{p}_1 & 0 \\ \bar{p} & 1 \end{bmatrix} * \begin{bmatrix} [R_r] & 0 \\ \vec{X}_r & 1 \end{bmatrix}$$

Since an entire family of planes is consistent with a given laser line measurement, additional constraints will be required to fully constrain the graph optimization. These additional constraints can come from: another laser line measurement taken from another point of view at a different height or pitch angle, a 3D plane measurement, or a weak synthetic prior factor. The use of both of these 2D and 3D mapping plugins together is shown in our paper Trevor et al. [2012] to improve mapping performance on experiments in multiple indoor environments. The 3D plane measurement sensor was a Kinect type 3D camera, which provides high resolution, but limited range and field of view. The 2D measurement sensor was an Hokuyo UTM30 laser scanner which has long range and wide field-of-view, but no vertical resolution. The experiments in this section describe how well these two measurement sources are able to work together to gather more map information.

2.4.6 Object mapping

In Rogers III et al. [2011], we developed techniques for mapping visual objects. This study and a novel technique called *semantic data association* will be presented in chapter 4. The plugin for handling objects in the *OmniMapper* is simpler than the plane and line plugins described above, because objects are tracked as 3D points.

$$h = R^T * (\vec{p} - \vec{t}) \tag{26}$$

The Jacobian with respect to the robot pose is then given by:

$$\frac{\delta h}{\delta X_r} = \begin{bmatrix} 0 & -h_z & h_y & 1 & 0 & 0 \\ h_z & 0 & -h_x & 0 & 1 & 0 \\ -h_y & h_x & 0 & 0 & 0 & 1 \end{bmatrix} \tag{27}$$

The Jacobian with respect to the landmark is given by:

$$\frac{\delta h}{\delta l} = R^T \tag{28}$$

2.4.7 Data association

Data association is the process of assigning sensor measurements to previously mapped or new landmarks. One of the core problems in SLAM is how to perform this data association, and how to recover from erroneous data associations.

There are a number of approaches to performing data association of measurements to mapped landmarks. The various techniques differ in what is needed to make the data association determination. The simplest technique, dubbed *Simple*, is not probabilistically motivated because it does not consider the uncertainty in the location of the robot relative to a measured landmark. A more sophisticated technique, called *Nearest Neighbor* uses the covariance between the robot and the landmark position to compute the probability that a measurement was generated from a mapped landmark. Finally, the most sophisticated technique, called *Joint Compatibility Branch and Bound*, performs probabilistic data association on all measurements simultaneously.

We have implemented these three data association algorithms within the *OmniMapper* via generic template programming. Each of these data association modules takes the current map and sensor measurements. Each module then returns indices referring to mapped landmarks for sensor measurements which correspond to these previously found landmarks. The data association module also inserts new landmarks for measurements which do not correspond to previously mapped landmarks.

The *Simple* data association module takes a measurement function which computes the error corresponding to inserting a new measurement on an existing landmark. This error function does not take into account the *shape* of the covariance

between the robot and this mapped landmark, and does not make a probabilistic data association. The error function is illustrated in equation 29. Here, a data association is accepted between pose x, landmark l, with measurement m if the predicted measurement between x and l is close to m. This data association module is fast and easy; however, it is not probabilistically motivated and can therefore not resolve fine detail on landmarks which are close. It also cannot handle loop closures well when uncertainty is too great.

$$e(x, l, m) = \|pred(x, l) - m\| \tag{29}$$

The *Nearest Neighbor* data association module does a much better job at resolving fine detail and performing data association in a loop closure. This is because the error function between a measurement and the robot is now the Mahalanobis distance between these entities using the covariance of the predicted measurement, as can be seen in equation 30. The covariance of the predicted measurement is computed from the joint distribution of the robot pose in question (usually the new robot pose) and the landmark being tested. This joint covariance is projected into the measurement space using the measurement Jacobians, as can be seen in equation 31. Here, $\Sigma(meas)$ is the covariance in measurement space, $\Sigma(x, l)$ is the covariance between the robot and landmark, and $\frac{dh}{dx}$ is the Jacobian for the function transforming landmark and robot positions into measurement space. $\Sigma(m)$ is the sensor error model. A nearest-neighbor data association is accepted if this Mahalanobis distance falls below a probabilistic threshold. It should be noted that a coarse version of the *Simple* data associator is used to filter and avoid testing data associations that are very far apart. This is needed due to the computational complexity of computing the joint distribution between the robot pose and each potential matched landmark.

$$e(x, l, m) = \|pred(x, l) - m\|_{\Sigma(meas)} \tag{30}$$

$$\Sigma(meas) = \frac{dh}{dx} * \Sigma(x, l) * \frac{dh}{dx}^T + \Sigma(m) \tag{31}$$

The *Joint Compatibility Branch and Bound* (JCBB) data associator extends the *Nearest-Neighbor* data associator with an additional joint compatibility test. Once the *Nearest-Neighbor* (or *individual compatibility* test) is passed, the group of measurements is considered together against a χ^2 test with as many degrees of freedom as measurements given. Measurements are added in an interpretation tree search with a branch and bound test. The branch and bound test prevents exploring other interpretations of landmarks and measurements which are worse than one that has already been tried. This module is based upon the paper Neira and Tardós [2001], which should be consulted for the implementation details.

2.4.8 Featureless mapping plugins

The feature-based mapping techniques described so far have been developed to operate in highly structured environments. They are useful for mapping man-made buildings, but are less useful in unstructured outdoor environments, or even highly cluttered indoor environments. In the presence of clutter, large sections of planes and walls might be hard to identify; to operate in these situations, mapping plugins based upon *iterative closest point*(ICP) algorithms were developed. The first of these is designed for indoor environments with a 2.5D planar constraint and lot of clutter; it is based upon the *canonical scan matching* implementation in Censi [2007]. The second of these is designed for outdoor environments where the robot may have general 6-DOF motion; it is based upon the 3D *generalized iterative closest point* (G-ICP) which was described in Segal et al. [2009]. The 2.5D mapping plugin based upon CSM is described in section 2.4.9, and the full 3D mapping plugin based on G-ICP is described in section 2.4.10.

Both of the featureless mapping plugins described in this section are based upon the iterative closest point (ICP) algorithm. The ICP algorithm is a two phase algorithm which is repeated until convergence. ICP algorithms take as their input two

point clouds (or scans), a reference cloud from the previous scan and a new cloud from the current scan. In the first phase, putative point matches are determined by selecting for each point in the new cloud the closest point in the reference cloud. In the second phase, these putative point matches form a set of constraints which is optimized via a nonlinear estimation algorithm to determine the relative pose between these two clouds. This procedure is now repeated by selecting a new set of putative point matches and solving for the relative pose until a convergence threshold is reached.

2.4.9 OmniMapper CSM

To enable operation in general indoor environments where no level of clutter or structural assumptions can be made, we developed a featureless map plugin called *OmniMapper CSM*. This mapping plugin uses the iterative closest point algorithm based on the point to line metric described in Censi [2008] to compute the relative pose between trajectory elements.

Andrea Censi provided a library routine called *canonical scan matching*(CSM) which computes the rigid body transform between two 2D laser scans. This library routine is called from two threads of computation in the OmniMapper CSM plugin. The first thread of computation processes incoming laser scans and registers them to the previously incorporated scan. The robot's odometry is used to provide an initial estimate of the transform between the two scans; the CSM library routine then completes the alignment to refine this transform.

The second thread of computation looks back at previous laser scans and finds loop closures. For each new scan, the centroid of the laser data projected as a point cloud is computed in a map coordinate reference frame. This centroid is compared to the centroids of all previous scans already incorporated in the map. Whichever is the closest to the current scan is then analyzed by the CSM library routine to attempt

to find the transform between these scans. If this procedure is successful, then a new pose constraint is added to the trajectory and the map is re-optimized.

The CSM library routine also computes an estimate of the covariance of the laser scan match using the technique described in Censi [2007]. This estimate takes into account the statistical noise of the laser scanner as well as the geometry of the environment when computing the covariance estimate. The result is a large covariance along a structure such as a hallway, with a tight covariance across the hallway, in the direction normal to the wall. This covariance estimate is used as the error model for each constraint added to the GTsam nonlinear factor graph.

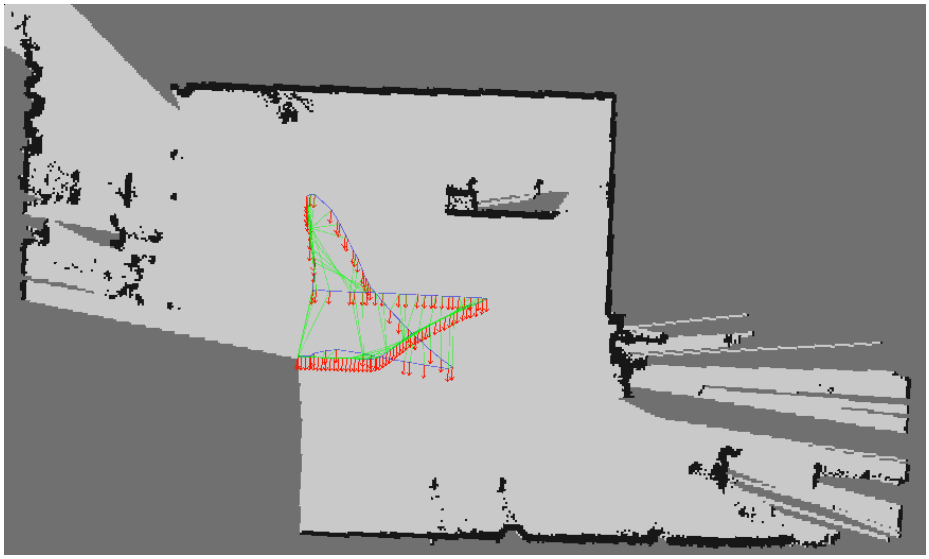


Figure 9: OmniMapper CSM builds a map of the Boeing lab at the MIRC building at Georgia Tech on the OmniX platform. The OmniX is a large holonomic industrial robot base. It has been augmented with Hokuyo UTM30 laser scanners for localization, mapping, and safety. The robot is also equipped with a drilling rig. OmniMapper CSM was able to achieve the 1cm accuracy needed to position the platform to perform a drilling demo.

OmniMapper-CSM was used to localize the Boeing Omnix platform in figure 9. In this experiment, the platform was able to determine its position within a few centimeters, which was needed to perform a drilling operation in an industrial setting.

2.4.10 OmniMapper ICP

The OmniMapper has also been extended with a plugin to use 3D iterative closest point (ICP) to build a detailed map in an arbitrary environment. The OmniMapper-CSM plugin described in section 2.4.9 is limited to indoor use because the planar ICP can only function well if the robot remains on flat ground. To support mapping arbitrary geography in outdoor navigation, a full 3D ICP mapping plugin was developed. This plugin uses the *Generalized-ICP* algorithm described in Segal et al. [2009] and provided in a *PCL* implementation.

The ICP plugin in *OmniMapper* contains two threads of computation. The first thread of computation processes new point clouds as they are produced from sensor data. These point clouds can be produced from many types of sensors. The sensor modalities which have been used with this plugin include Hokuyo UTM30 laser scanners which are mechanically actuated via Directed Perception pan-tilt units, Kinect type 3D cameras, and Velodyne 32E 3D LIDAR scanners. The second thread of computation looks for loop closures with the most recent point cloud and other point clouds which were already used. These two threads provide pose measurement constraints which are used to compute the robot's trajectory. This trajectory information is used to render all point cloud data into one large point cloud representing the entire map.

The *Generalized ICP* algorithm described in Segal et al. [2009], and provided as an implementation in *PCL*, has a number of advantages over the vanilla ICP procedure described above. Due to limited sensor resolution, exact points in the reference cloud are rarely imaged by the new cloud. In the case of a long range sensor with limited (vertical) resolution such as the Velodyne 32E, the reference cloud points could fall in-between the new scan. These points could be up to a meter apart. Due to this effect, the errors are projected onto the surface normals to constrain their correction only in the direction of the normal. This is called the *point-to-plane* error. In Generalized

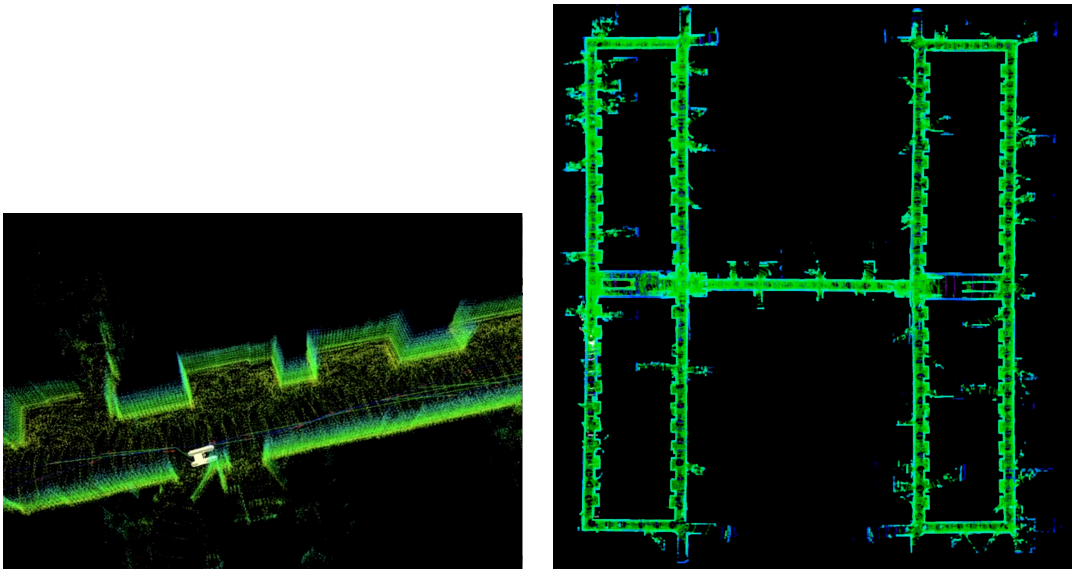
ICP, both the new and the reference cloud normals are used in computing the point-to-plane error; this technique significantly improves performance as is shown in Segal et al. [2009] and empirically verified in our own comparisons.

The ICP algorithm is costly in terms of computation time. To maintain online operation, the resolution and density of the point clouds is reduced to a more manageable level. The filter which contributes the most to producing manageable point clouds is a voxel grid filter. This filter limits the density of the point cloud to one point in each 5cm voxel. This reduces the density close to the robot without affecting the long range density, where errors are more apparent and can be used to correct the robot's trajectory.

Even with reducing the density of the incoming point clouds, the ICP routine requires almost 2 seconds per point cloud. Fortunately, since the robot has a good estimate of its relative motion from IMU and odometry, the ICP routine can be initialized close to the final solution. This allows for significant platform motion between point clouds; therefore, the robot can still travel quickly and keep track of its motion and build a map. In figure 10(a), a robot is seen using OmniMapper-ICP to build a 3D map of a hallway in an office building.

When the robot crosses its previous trajectory, a second thread looks for possible loop closures by trying to perform ICP between the point clouds observed at these locations. If these point clouds can be aligned, then the relative pose measurement factor is added to the graph and the map is re-optimized. A simple case where the robot was able to use this loop closure routine successfully is shown in figure 10(b). Here, the loop closure routine worked well because the robot crosses its path several times in moving through these hallways. This loop closure routine is unlikely to work if the robot has traveled a long distance before returning to this previous location along the trajectory. This is due to the fact that ICP routines must be initialized within a relatively close region of convergence, approximately 0.5 meters. This can

be made to work with the help of a GPS plugin, as can be seen in a large scale map in figure 11(a). Briefly, the GPS plugin adds pose constraints along the trajectory with error models given by satellite fix quality estimates. OmniMapper-ICP is used together with the GPS plugin for very large outdoor mapping, as seen in figure 11(b).



(a) A 3D view of a small segment of hallway from OmniMapper ICP

(b) The final indoor hallway map built from OmniMapper ICP

Figure 10: OmniMapper ICP run on an indoor office environment. Measurements are made by a Velodyne 32E 3D laser scanner from a ground robot.

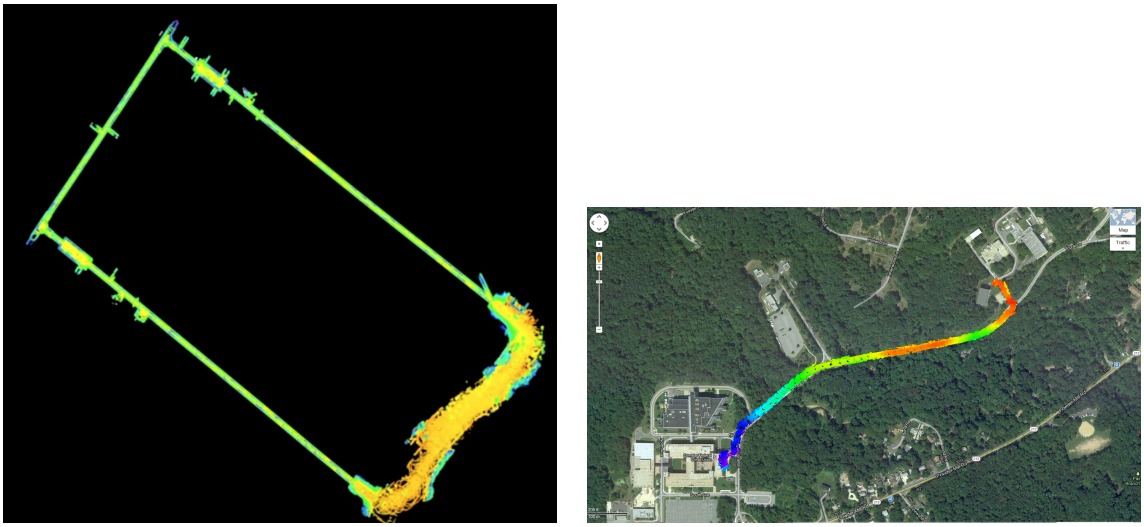
2.4.11 Navigation cost-map generation

Navigation in a cost map, how cost maps are formed.

2.5 Effects of sensory precision on map performance[5]

The current equipment needed to build a domestic service robot is too expensive for the average family to afford. Recent developments in sensor technology such as the Microsoft Kinect 3D camera have been revolutionary in delivering a new sensor modality at a commodity price. Current research sensors such as LIDAR systems are very precise and expensive units. This study will address how feature based mapping

[5] ***This section is based upon Rogers et al. [2010]***



(a) A path through an underground tunnel over one kilometer in length. The robot started outside in the fuzzy area in the bottom-right and entered the upper tunnel, proceeding in a counter-clockwise direction. GPS measurements helped make this loop closure as the robot exited the lower tunnel in the bottom-right portion of this map. GPS was sufficient to correct the trajectory enough that ICP was able to close the loop.

(b) A robot uses Omnimapper-ICP together with the GPS plugin to map an outdoor route. Robot point cloud data is shown along optimized trajectory overlaid on Google Maps.

Figure 11: OmniMapper-ICP used with GPS plugin to enable operation over large distances in outdoor and mixed indoor/outdoor environments.

techniques using *OmniMapper* can allow for the use of less precise, and therefore cheaper, sensor equipment on mobile robots.

Making robots cheaper also enables new uses in dangerous environments such as disaster sites or urban counter-insurgency operations. Simultaneous Localization and Mapping (SLAM) techniques can be used to build maps which can be shared with emergency workers or soldiers as robots complete exploration missions [Berrabah et al., 2010]. Projects such as the Micro Autonomous System Technologies Collaborative Technology Alliance (MAST CTA) [ARL, 2006] are working to develop techniques and components to enable robot mapping with low power and small form factor robots and sensors. MAST's mission is to develop robots which are autonomous and collaborate to provide situational awareness in urban environments for military and rescue operations. MAST platforms are required to have a small form factor and

long operational availability, so we must understand where compromises can be made on sensor performance.

There is a need to determine what level of performance is required from sensors to enable SLAM techniques to deliver a map of sufficient quality from autonomous exploration of an unknown environment. Sensors such as laser scanners are appropriate in laboratory applications but other sensory modalities such as radar or vision may be preferred when considering cost levels, power usage, and size. Some of the accuracy, density, annular confinement, and field of view assumptions which are common with laser scanners might no longer be valid with these other sensory modalities. Low cost radar based scanners are currently under development as part of the MAST project; until these are available, we will simulate the performance of radar in software using data from laser scanners. To investigate the effects of various properties of a sensor on the mapping result, we use high quality laser data and degrade properties such as the maximum range, angular resolution, field-of-view, or range accuracy. We call this *de-featuring* the data. The performance requirements for robot mapping must be determined to guide the development of these new miniaturized radar scanners to ensure that they will work well for robot mapping tasks.

An objective benchmark for comparing SLAM algorithms was reported in Burgard et al. [2009]. This benchmark compares the incremental relative error along the robot's trajectory instead of comparing maps. This allows the authors to directly compare the results between different algorithms or sensor types. Ground truth is generated by manually aligning laser scans using ICP as an initial guess. We use the relative pose error metric and ground truth generation method from this benchmarking paper to assess SLAM performance levels under various sensor configurations.

Research has been reported on analysis of localization performance and sensing characteristics. O'Kane and Lavalle proved that it is possible to localize in a known map with a robot equipped with only a contact sensor and a compass in O'Kane

and LaValle [2005] and Jason and LaValle [2007]. The authors determined that it is impossible to localize a robot in a known map if the compass is replaced with an angular odometer. In the SLAM domain, work has been done to support simpler and less expensive sensor modalities. Bailey developed techniques for delaying initialization of landmarks in bearing-only SLAM in Bailey [2003]. A paper by Müller *et.al.* relates the performance of their 6D SLAM algorithm to the precision of their 3D laser scanner in Müller et al. [2006].

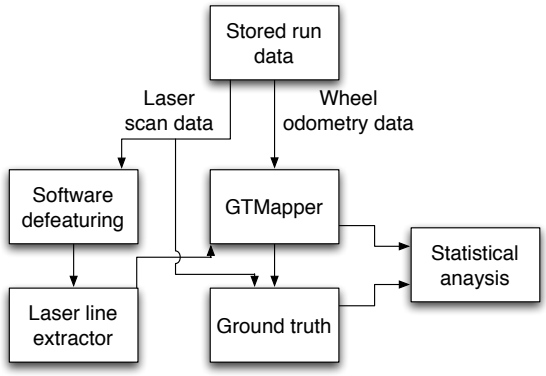
The feature detectors developed for this mapping application have been designed to be robust to noise and performance degradation. We have chosen to extract line segments from laser range data using the RANSAC technique described in section 2.4.2. We have selected parameters such as minimum line length and maximum gap which correspond to the size of walls and door openings in a typical office environment.

The experiments in this section were run on data collected from two types of mobile robots and were processed offline to compare mapping performance with different sensor de-featuring. Though these techniques were compared in offline operation, it should be noted that the mapper can comfortably run online on a modest CPU.

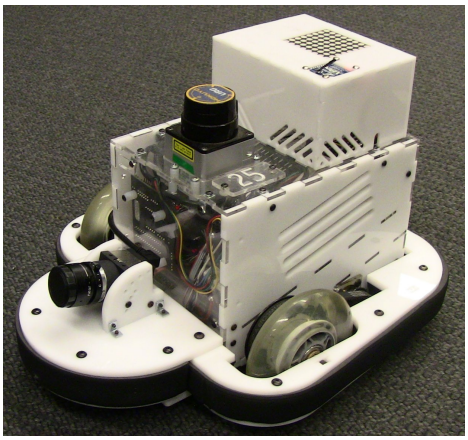
2.5.1 Experimental Design

The operation of the experimental framework is shown in figure 12(a). Ground truth was generated by manual alignment of the full unaltered laser scan data in an initial run. This serves as an adequate estimate of ground truth for the purpose of this relative comparison of path trajectory error with different sensor performance levels. It allowed the operator to correct the (x, y, θ) between subsequent poses so that the laser scans came into alignment. This information was then captured in a ground-truth file where it was compared to the posterior relative displacements in the test run trajectories.

Stored robot data was replayed with various settings of laser scan data de-featuring. The laser line extractor extracted line segments from the de-featured laser data and sent them to the mapper. Once the run was completed, the robot trajectory from the mapper was compared to the ground truth for this run. The average incremental absolute error was then recorded.



(a) The experimental setup



(b) The Scarab mobile robot developed at the University of Pennsylvania

Figure 12: Experimental setup and equipment used to gather data for robot experiments

2.5.2 Experiments

The first source of sensor data used in these experiments was a series of logs taken from the *Scarab* robots developed at the University of Pennsylvania. An image of the Scarab robot can be seen in figure 12(b). The Scarab is a differential drive mobile robot equipped with a Hokuyo URG laser scanner. In this series of experiments, the robot was tele-operated for ten runs in an indoor office environment to collect data which is used for offline processing.

The laser scanner accuracy was de-featured for our analysis to determine what level of performance in the mapping task can be expected with progressively simpler (and therefore less expensive and lower power) sensors. All sensor values were captured

during the data collection phase and were de-featured with software in the post-processing phase.

In the first round of experiments, several de-featuring modifications were made to the laser scanner readings. First, the range of the scanner was restricted. Second, the angular precision of the laser scanner was limited by omitting beams from the line extraction process. Thirdly, the angle subtended by the laser scan was reduced. Finally, the range accuracy of the sensor was corrupted by varying degrees of Gaussian noise. This series of experiments will establish which parameters of the scanner can be varied while still yielding sufficient performance in the mapping task.

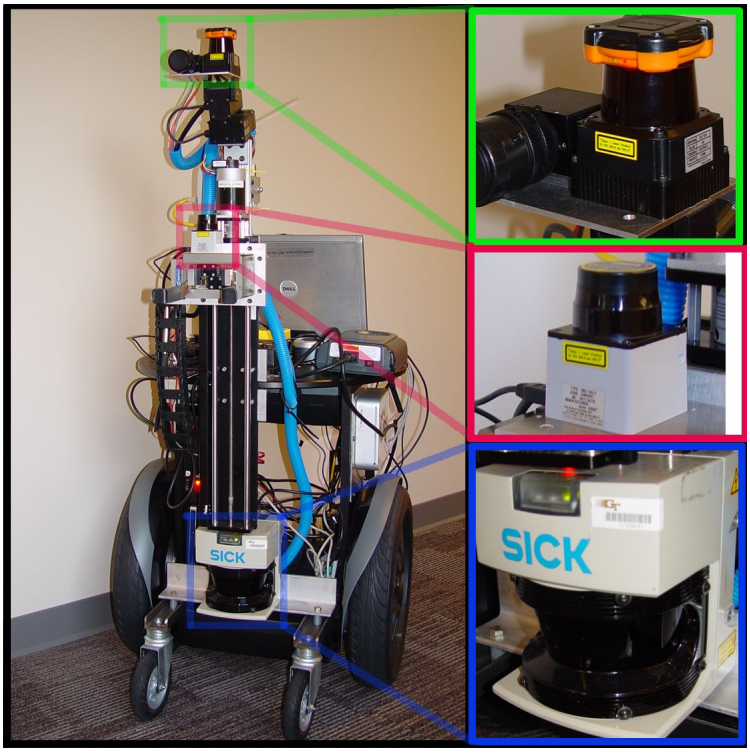
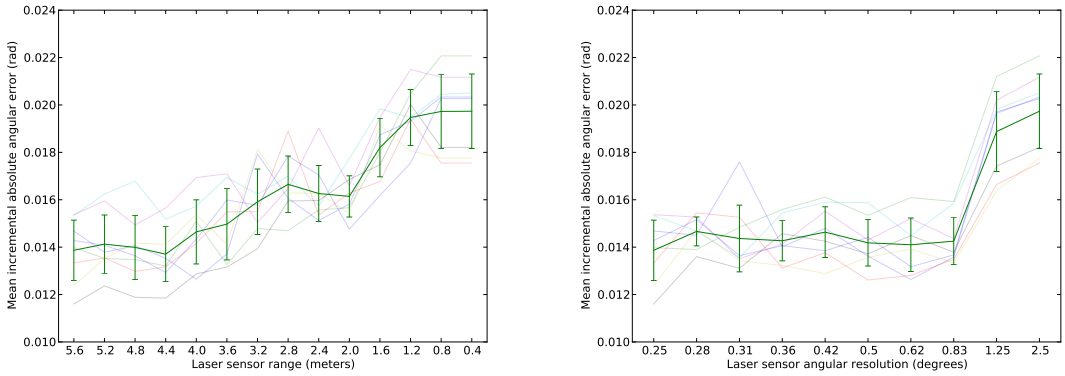


Figure 13: The Georgia Tech robot "Jeeves". Left: Full robot platform. Right-top(green): Hokuyo UTM-30 laser scanner. 30 meter maximum range, 270 degree viewing angle, 0.25 degree angular resolution. Right-middle(magenta): Hokuyo URG laser scanner. 5.6 meter maximum range, 270 degree viewing angle, 1 degree angular resolution. Right-bottom(blue): SICK LMS291 laser scanner. 80 meter maximum range, 180 degree viewing angle, 1 degree angular resolution.

In the second experiment, the Georgia Tech robot "Jeeves" was outfitted with

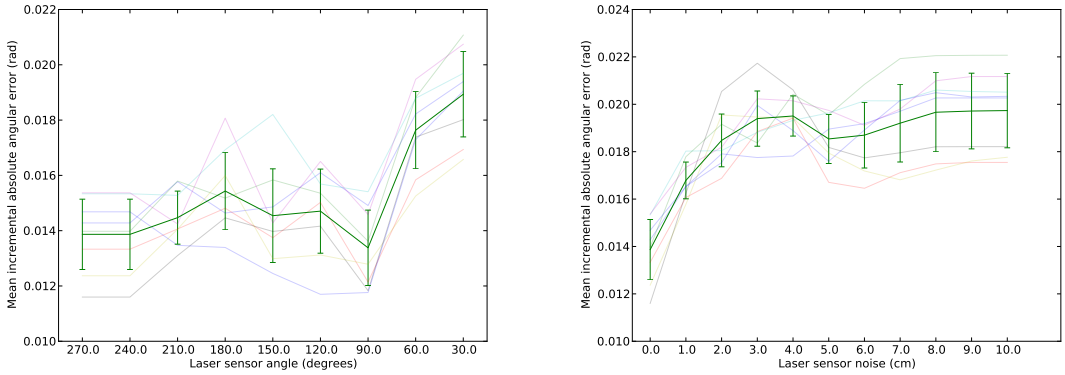
three different laser scanners to collect a run where each laser can be used to compare the mapping results. The robot is a Segway RMP 200 modified to be statically stable, as seen in figure 13. This platform is configured to simultaneously collect data from the SICK LMS291, Hokuyo URG, and the Hokuyo UTM 30 laser scanners, all of which are seen in the right of figure 13. This data is used individually in the mapping process and the resultant relative errors are compared. In this experiment, there is no need to de-feature the performance of the laser scanners in software since each exhibits a unique price and performance level.

2.5.3 Results



(a) Maximum range restricted between 0.4m to 5.6m

(b) Angular resolution restricted between 0.25° to 2.5°



(c) Angle subtended by laser restricted between 270° to 30°

(d) Gaussian noise added with variance up to 10cm

Figure 14: Mean incremental heading error with various types of sensor *defeaturing*

In the first set of experiments, Scarab robots were driven along a similar trajectory in an office building 10 times. Two of the runs were of poor quality due to wheel slip on a thick concrete seam and were eliminated from the analysis. In figure 14(a) the mean trajectory error is displayed from a series of test runs where the maximum range of the laser scanner was varied from 5.6 meters down to 0.4 meters. 5.6 meters is the maximum range of the Hokuyo URG laser scanner. It can be seen from this figure that the performance is poor when the range is restricted to below 2 meters, but it rapidly improves at longer ranges to a nearly constant level of performance. The residual incremental error (due to the inaccuracy in the odometry of the robot) is reached as the range is restricted to 0.8m and below. This experiment indicates that the accuracy of the mapper can be maintained until the range of the laser scanner is restricted to about 1.6 meters, roughly the width of the hallways in the test site.

In figure 14(b) the mean trajectory error is displayed from a series of test runs where progressively higher proportions of the laser beams are left out of the line extraction process. This test is meant to simulate a sensor with less angular resolution than the Hokuyo URG. It is apparent from the graph that until the resolution is lowered to around 1 degree, the mapper is able to deliver a similar level of performance to the results from the fully featured laser scan. This corresponds to throwing away 75% of the data coming from the laser scanner. When the resolution is lowered below 1 degree, the accuracy drops to the level of the robot odometry alone.

In figure 14(c) the mean trajectory error is compared with various angles viewed by the laser scanner. The default Hokuyo URG laser scanner can view 270 degrees. This metric is meant to simulate a sensor which views a smaller angle. The viewing angle is always faced towards the front as it is expected that the sensors would need to observe this region in order to avoid hitting obstacles during autonomous operation. This graph suggests that the mapper remains accurate until as little as 90 degrees is viewed from the laser scanner. This is probably due to the fact that in this test the

range was kept at its maximum value of 5.6 meters, so the mapper was still able to see wall line segments. If the range was restricted while the angular view was also restricted, then we would expect poor performance.

In figure 14(d) the mean trajectory error is displayed from a series of test runs where Gaussian noise is added to the range measurements. The noise is sampled from a Gaussian distribution with the variance indicated along the x-axis. It can be seen from this figure that with even a moderate 2cm noise is added to the laser scan significant error is introduced into the map. The mapper is particularly susceptible to noise in the laser ranges, though it should be noted that this noise level is significantly higher than in a typical laser scanner.

Error	URG	SICK	UTM-30
Position error	0.020162	0.020942	0.009785
Angular error	0.009924	0.006612	0.007033
Resolution	1°	1°	0.25°
Range	5.6m	80m	30m
Range Error	1%	$\pm 35mm$	$\pm 30mm$
Angle	240°	180°	270°

Figure 15: Absolute incremental position and orientation error for test comparing performance of mapping with three typical laboratory laser scanners

In the second experiment, mapping results from three different laser scanners which are commonly used by the SLAM research community were compared. An example map from this run is shown in figure 16. The robot successfully closed the loop with each of the laser scanners. The absolute relative incremental position and angular errors can be seen in table 15. It can be seen with these results that the SICK and URG mapping results are similar in displacement error, whereas the UTM30 is much more accurate in recovering the relative displacements. This is to be expected since the UTM30 slightly outperforms the SICK 291 laser scanner in angular resolution, range accuracy, and speed.

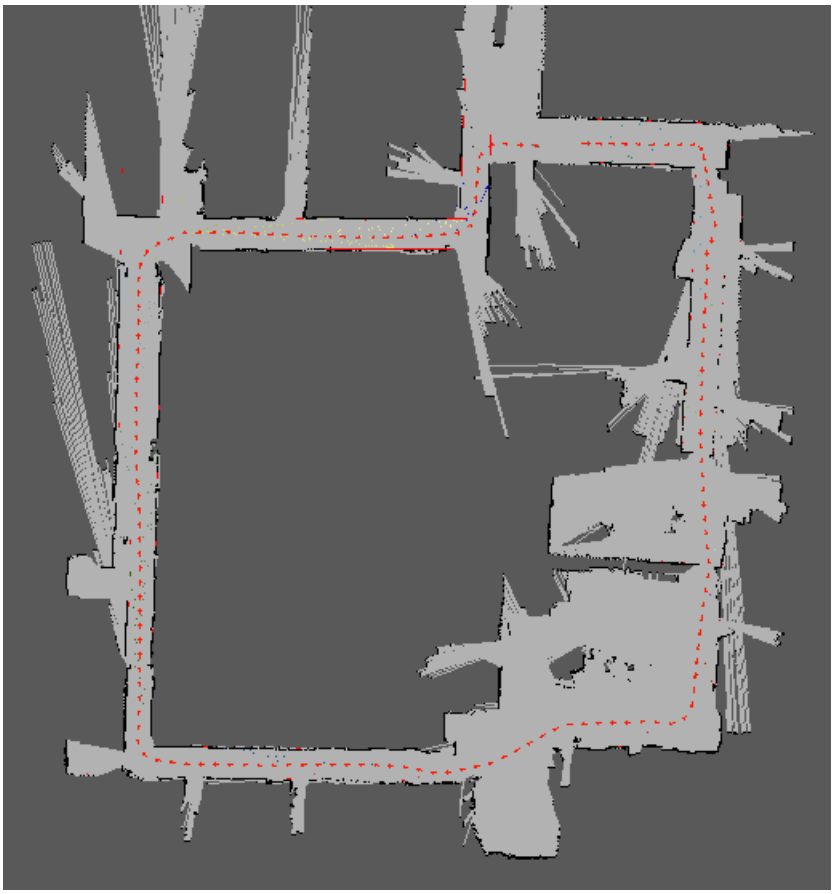


Figure 16: The map of the the corridors in our lab. There are two large storage rooms in the lower right. The map is shown as an occupancy grid only for display purposes – all measurements are made based on wall features.

2.5.4 Discussion

The longer range of the laser scanner does not appear to improve performance in the office environment. This indicates that the line extraction algorithm is working well when even a small amount of the wall can be seen from the scanner, so the range of the scanner doesn't need to be much more than the width of the hallway in order to achieve good performance. We believe that in general the range of the scanner needs to be no more than such that it can see about 1 meter of the wall from the middle of the hallway.

We were surprised to find that the mapper was so susceptible to range accuracy given that our line extraction algorithm performs a least-squares fit to many laser

points and should be able to handle zero-mean noise. It should be noted that the noise levels explored here were relatively large and that the line extraction performs very well for noise levels more typical of laser scanners. We believe that, with additional parameter tuning, the line extractor could be made to perform well with more noise; so this technique should translate well to radar scans.

The angular resolution parameter is effectively reducing the range of the scanner slightly because the missing resolution is causing gaps in the extracted lines beyond a certain range. This prevents long lines from being extracted; however, we have already determined that range much longer than the hallway width is not very important in the range test. We have now also determined that high angular resolution is not needed to perform the mapping task well.

For the second experiment with the robot "Jeeves", we notice that the incremental angular error is comparable between the SICK and UTM30 laser scanners, with the URG performing slightly worse. We believe this is because the office environment for the second experiment has larger, cluttered rooms for which the limited range would often result in few measurements of the walls with the URG. The incremental displacement error seems to show that the UTM30 is performing better than the SICK laser scanner. In our first experimental analysis we determined that the mapping results are particularly susceptible to range accuracy, but the The UTM30 claims a $\pm 30mm$ range accuracy whereas the SICK claims a $\pm 35mm$ range accuracy. There is only 5mm difference here so this is likely not the main cause. It is more likely that the extended viewing angle combined with the longer range of the UTM30 is allowing the robot to observe both the walls behind it as well as those in front of it for a significant portion of the run. The SICK can only see in front of the robot and is therefore only correcting based on the walls in front – it cannot see the wall behind the robot as it passes through a doorway. We think that this effect was not seen in the first experiment with the Scarab robots because they were operated in an

environment with narrow corridors and few larger rooms. The Scarab robots never had the opportunity to pass through doorways and therefore could not benefit from this effect.

Based on these results, we determined that the Hokuyo UTM30 is the ideal laser scanner for use in environments with larger open spaces. It has additional advantages over the LMS291 laser scanner, namely size and power usage. In environments with more confined spaces, the Hokuyo URG performs well enough to make useful maps.

2.6 Discussion

This chapter presented the mobile robot mapping components which were developed for this thesis. We have developed algorithms which can be used by a mobile robot to automatically segment maps into useful components based upon shared information between landmarks. Landmarks share information because they are observed together; this is an automatic way of segmenting rooms in a domestic environment.

We have also contributed an EM-based algorithm for detecting and correcting errors in data association, a core issue in reliable application of SLAM in real-world situations. This section also introduced a version of object-based SLAM where data association is provided not by geometric arrangement but by high level reasoning about object identity. Further developments on object-based SLAM will be detailed in chapter 4.

These first two studies used a variety of mapping algorithms and custom implementations. We produced a mapping library called *OmniMapper* and a series of plugins which can perform feature-based and featureless mapping on many types of sensor input. This library is designed to be extensible to work with new types of measurements and landmarks in the future. This library is used by our lab as well as some of our project partners in the Army Research Laboratory. We are also preparing for an open-source release of this library.

Finally, we looked into how we can reduce the complexity and cost of some of these robot platforms by simplifying their sensor suite. This study could be used to determine which aspects of sensory performance are able to be traded off with limited consequences to robot mapping performance. In addition, the most common sensors in use today were compared on a mapping task.

The mapping components described in this chapter will be further extended for multi-robot mapping in chapter 3. Additional techniques for mapping with objects and object recognition and classification algorithms will be discussed in chapter 4. A framework for combining place recognition and object mapping for semantic SLAM is discussed in chapter 5.

III

MULTI-ROBOT MAPPING

Multi-robot systems have a number of advantages over monolithic robot systems such as redundancy, reliability, availability, and heterogeneity. A robotics application handled by only one robot is subject to single point failures. With redundant and compatible multi-robots such failures can be overcome. Multi-robot systems can often be useful for tasks such as mapping an unknown environment; these tasks can be completed more quickly because area is covered in parallel. Heterogenous teams of mobile robots can even use components which compliment their team-mates without reducing reliability.

It is unlikely that just one monolithic robot will work by itself in the home. The current household already incorporates many automated appliances which expedite domestic tasks such as the dishwasher, the washer, and the dryer. In the future, these components will work together with domestic service robots in a network to perform chores. In addition to typical home automation merging into a multi-robot system, the mobile robot component could be realized through a small set of simpler robots working together to accomplish tasks instead of one robot working alone. Various tasks such as meal preparation consist of sub-tasks which can be completed in parallel by a team of robots as in Beetz et al. [2011]. Other tasks such as moving furniture are easier for a team of n robots each with $\frac{1}{n}$ the strength of an alternative single robot approach as in Rus et al. [1995].

Mobile robots are already widely used by first responders both in civilian and military operations. Today the operations are largely through tele-operation. Such a mode of operation challenges the operator as the cognitive load is significant [Zheng

et al., 2011]. There is consequently a desire to introduce some degree of autonomy to reduce the burden on the operator. For design of fully autonomous systems there is a need to supply the system with a complete map of the environment or to endow the system with methods for automated mapping and exploration.

Moving from single robot systems to multi-robot teams poses a number of additional challenges. First of all the operator is posed with an added complexity in terms of controlling multiple entities at the same time. In addition, integration of maps generated by multiple robots into a coherent representations is known to be a challenge. Finally, there is a need to consider how the team-members can cooperative to optimize the exploration of a previously unseen environment. There has been some progress reported on multi-robot mapping as presented in Fox et al. [2006]. A number of methods for exploration of spaces have also been presented, see Parker [2008] for a recent summary of related research.

3.1 Background

Multi-robot mapping based upon the *M-space* representation from Folkesson and Christensen [2004] was presented in Benedettelli et al. [2012]. The authors demonstrated how this representation can be used to merge maps across pairs of robots when they rendezvous and establish their relative pose. Our approach to multi-robot mapping is to form the global map on a remote server based upon an initial estimate of the robot's relative starting poses.

Multi-robot mapping and exploration was addressed by Fox et al. [2006] and Vincent et al. [2008]. These papers build a map using up to 3 robots with a decision-theoretic planner that trades off robot rendezvous operations with frontier exploration. These robots rendezvous to determine their relative pose transforms to provide constraints to recover the final map.

In Olson et al. [2012], the authors describe a system which controls a team of up

to 14 mobile robots in an urban reconnaissance mission. In this system, a planning algorithm allocates robots to explore in an unknown environment and build a map.

In Hollinger et al. [2010], the authors prove performance characteristics on a multi-robot collaboration strategy to perform adversary search. By representing the topological configuration of a map as a graph, the robots can guarantee that the adversarial search will prevent re-contamination of previously cleared nodes with an arbitrary sized team. In Joyeux et al. [2009], the authors describe a distributed system for managing robot plans for performing high-level tasks. This architecture prevents conflicts between robot plans and can handle communication failures. These papers both present strategies and architectures for collaboration between robot agents to perform tasks.

One of the main concerns in building a map across a team of robots is how to exchange information efficiently between agents. The first study, in section 3.2, will address this with the use of *distributed data fusion*. This study also addresses how robots can coordinate their actions via a recruiting strategy.

Heterogenous teams of mobile robots can work together to perform a variety of tasks efficiently. A study which fuses maps built from two robots with different sensing modalities is presented in section 3.3. With this multi robot mapping system, a robot with 2D mapping sensors is able to assist a 3D mapping robot in making a more accurate map.

When larger teams of mobile robots work together to build a map of an unknown environment, the strategy they use for collaboration can have an impact on the effectiveness of the team. Some resources can be dedicated to work together with other robots as they explore, so they are available to handle T-junctions and branching. Three strategies for collaboration are compared on mobile robot exploration and mapping in section 3.4.

3.2 Autonomous 2D mapping with a team of mobile robots[1]

Entering an unknown environment poses a challenge in many different respects. When soldiers or first responders experience such a situation, it is stressful as there is great uncertainty. What is behind the next corner: explosives, gas, or another human? It is these situations which result in the most casualties. There is consequently an interest to consider how robotics technology can be utilized to create a map of an environment before humans enter it. It is particularly of interest to study how a team of small robots can cooperate to build a map of the environment. To accomplish this task, a number of issues must be addressed; i) detection of important structures in the environment, ii) integration of structures into a consistent map, iii) integration of maps across the team of robots, iv) autonomous exploration of the environment to ensure full coverage of the area of interest, and v) communication of the map to an end-user.

Consider the scenario shown in figure 17. A robot is called in to provide mapping. As it enters the building it returns imagery that specifies the presence of a staircase. At the top of the staircase is a corridor with multiple doors. The corridor is lit, which might indicate the presence of humans. The information is extremely valuable for the soldiers entering the building. This is the motivation for the research presented here.

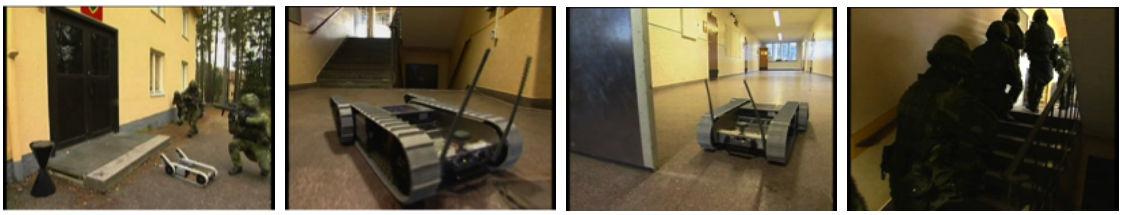


Figure 17: The example scenario that illustrates the value of using robots for early reconnaissance

The scenario considered here is an indoor office or other domestic environment,

[1] ***This section is based upon Rogers et al. [2011]***

which is characterized by planar walls and engineered doorways. Consequently, walls are described by line segments (section 2.4.2) and doorways are described as an position, orientation, width and height, and detected by co-incidence of laser data and vertical edges (Section 2.4.3). Detected structures are integrated into a coherent map using a graphical model (Section 3.2.2.2). The maps acquired on each robot are exchanged and fused into a coherent representation (Section 3.2.2.3). As a robot explores the environment it may encounter an intersection (multiple hallways/rooms). At such intersection points the robots will request assistance from other robots to explore the corridor that it cannot explore itself (Section 3.2.2.1). In reality the exploration strategy is a distributed depth-first graph search method and it is guaranteed to cover the full graph.

Exploration and cooperative mapping is by no means a new problem. Early work was presented in Almeida and Melin [1989]. Frontier based exploration was presented by Yamauchi et al. [1999]. Graph based exploration has been presented by Dudek et al. [1991] and Choset and Burdick [2000]. Topological mapping and exploration has also been proposed byKuipers and Byun [1987]. The main innovation here is the distributed exploration and map fusion for multiple cues.

In Section 3.2.1 the basic setup for the investigation of mapping and exploration is presented both in terms of the systems design and the environment. The methods used for the integration of the system are presented in Section 3.2.2. Results from the joint experiment are presented in Section 3.2.3 and issues for future work are presented in Section 3.2.4.

3.2.1 The experiment

3.2.1.1 The environment

The joint experiments defined in a collaboration with the Army Research Laboratory (ARL) are setup as scenarios to explore different aspects of design of micro

autonomous systems technology. The scenarios are not considered to be "demonstrations", but instead as organization of a system for a range of experiments over time. The experiment 3.1 is considering exploration and mapping of indoor environments. The prototypical scenario has multiple corridors and offices. The future versions of this experiment will include space across multiple floors. The scenario chosen here is one of the floors in the Computer Science Building at University of Pennsylvania. The floor map and example images are shown in figure 18.

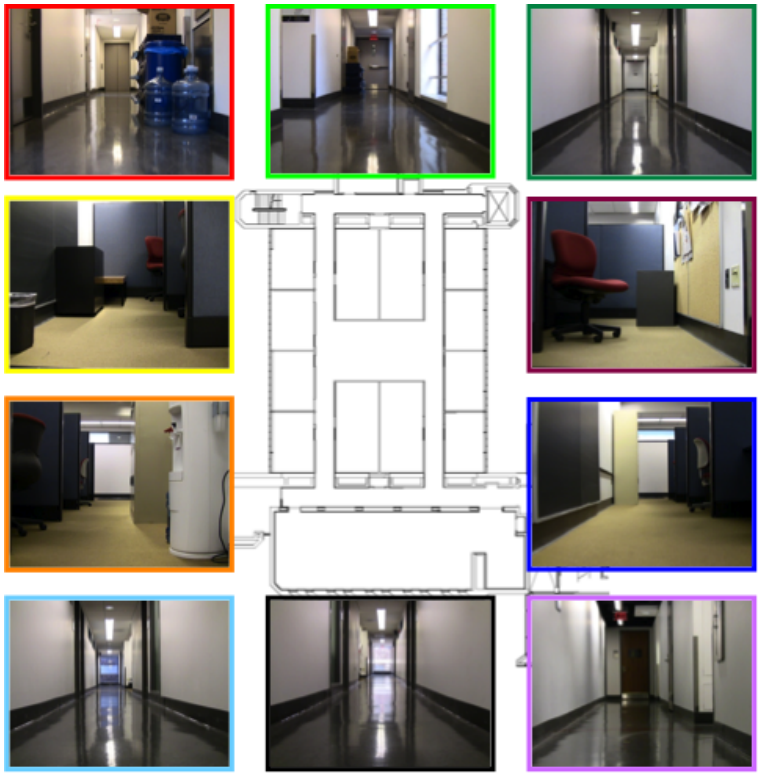


Figure 18: Floor map and example images for the experiment

The environment has two loops and several blind alley type corridors, which is ideal for topological based exploration/mapping experiments. There are relatively few features in the corridors, which would challenge visual localization and mapping but is acceptable for laser/vision based localization and mapping.

3.2.1.2 Platforms

For the evaluation of the methods considered here (See Section 3.2.2), a number of Scarab platforms are used. The basic Scarab robot is a differential drive platform that is 25x25x20 cm in size. The robot has an on-board PC with a 1.5 GHz pentium, 1GB RAM, and a 32 GB SSD disk. The robot is running standard Debian Linux. For communication each robot is equipped with a 801.11s (ZigBee) network, which is a standard mesh network with a limited range to adequately emulate limited range communication challenges. Each robot is equipped with a USB camera, a Hokuyo URG-4x laser scanner and wheel encoders. A team of Scarab robots were used for the experiments. A picture of the robot team is shown in figure 19.

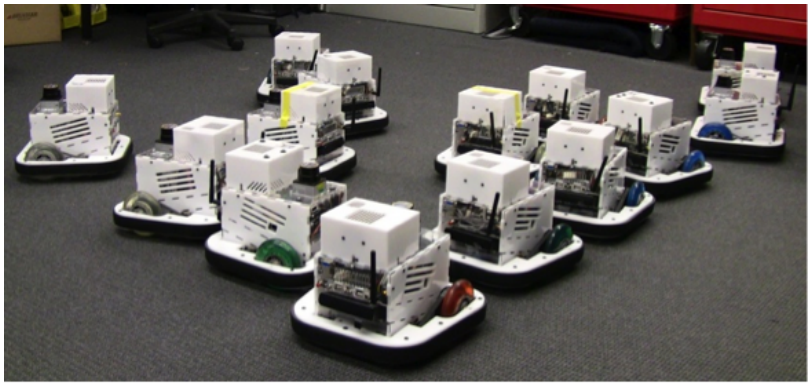


Figure 19: The team of Scarab robots used in the experiments.

3.2.2 METHODOLOGY

3.2.2.1 Exploration strategy

Efficient and complete coverage of the target structure is necessary to build a map for use by human soldiers for clearing structures or first responders in a disaster recovery. Our current focus is on structured indoor environments, such as an office or other domestic building. The current strategy has been developed to provide complete coverage in a target structure by coordinating exploration among multiple robots while maintaining robustness to communication loss or (partial) hardware failure.

For the single robot case, the robot starts in "Corridor Following Mode". In this mode, laser scan data is analyzed to find gaps of sufficient size to be classified as a doorway or corridor. The robot selects the candidate gap which is closest to its current direction of travel and records the location of other gaps for future exploration. When the robot reaches a dead-end, there will be no candidate gaps for exploration in the current laser scan. If the robot has recorded unexplored gaps, then the robot selects the nearest unexplored gap and backtracks along its path. When the robot reaches the nearest unexplored gap, it marks it as explored and resumes "Corridor Following Mode".

The single robot exploration is extended to multiple robots while handling communication loss and partial hardware failure through a simple protocol. When a robot sees a candidate gap, in addition to recording the location of this gap it also broadcasts it to the other robots. If the communication fails, then those robots to which communications have been lost will not know about this gap. When any robot reaches a dead-end, it will consider all of the candidate gaps that it has personally observed as well as those which it has received from other robots. The robot will then enter an auction by announcing a bid for the nearest candidate gap. This auction lasts for 5 seconds – during this time any other robot which reaches a dead end may also bid for the same candidate gap (or any other). Each robot compares its bid to the bids it receives from the other robots, and each one determines which robot has won the auction. If communications fails during the bidding process, then multiple robots can determine that they have won the auction for this candidate gap. This is not a problem because this location will be explored by multiple robots instead of just one; it is only a less efficient use of resources.

When a robot arrives at a candidate gap, it announces this fact to the other robots, who then delete this candidate gap from their records. If communication fails, then the other robots will simply not know that this robot has explored the candidate

gap and will eventually re-explore it themselves. If the robot is disabled or destroyed before reaching the candidate gap, then it will not announce that it has arrived and another robot will eventually explore this gap.

The overall coordination strategy is illustrated in figure 20

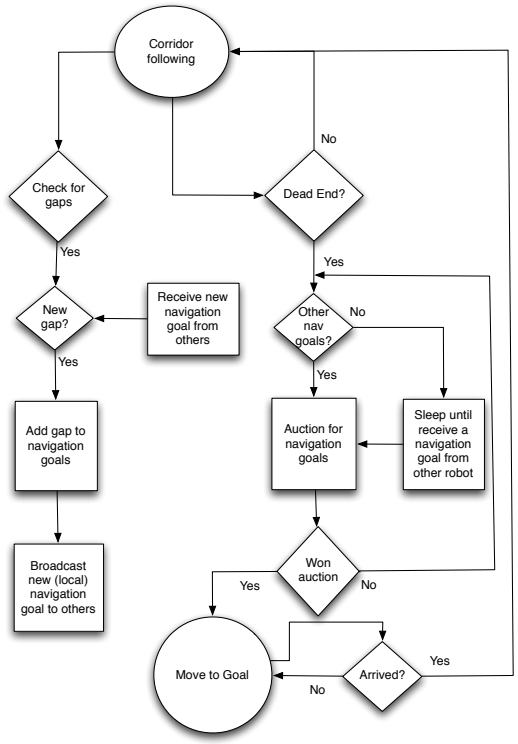


Figure 20: The exploration / coordination strategy used as part of the experiment

3.2.2.2 Local mapper

The local mapper used in this study on each robot is based upon the *OmniMapper* library described in section 2.4. The local mapper uses the map plugins for 2D walls described in section 2.4.2 and doors described in section 2.4.3. The robots used in this study provide odometry data and measure doors and walls in the environment.

Robot odometry is used to establish initial conditions for the location of new structures in the environment as well as to provide a loose string of measurements extending through the environment. Without these measurements, the robot would

be unable to determine motion in a hallway with smooth walls and no doors. When a new door or wall measurement comes in, the robot odometry is interpolated to find where the robot was when this measurement was taken. A relative pose measurement is inserted between the last recorded pose and the pose when this measurement was taken. Data association is performed between this door or wall measurement to other mapped structures from this new pose. At the time of this joint experiment, this data association procedure was a simple nearest neighbor search. Currently we have developed both a Mahalanobis distance based data association and a joint compatibility branch and bound based data association to increase data association reliability. If the new measurement does not data associate to a mapped feature, then a new feature is inserted in the map. The measurement is applied between the new pose and the data associated feature, and the entire map is re-optimized.

3.2.2.3 Distributed-Data-Fusion[2]

To perform robust multi-robot mapping, we introduce DDF-SAM, an implementation of Distributed Data Fusion (DDF) [Durrant-Whyte and Stevens, 2001] in a Smoothing and Mapping (SAM) context. The goals for the distributed system are to be resilient to node and communication failure, while minimizing both local computational and communication cost. As a motivating example, consider a naive solution to building multi-robot maps in which each robot sends all measurements Z of the environment to each neighboring robot. Each robot, in this case, then builds a full factor graph over all robot poses X from all robots and landmarks L, and performs nonlinear optimization over the entire system to minimize the measurement prediction error $E = \frac{1}{2} \|h(X, L) - Z\|^2_\Sigma$ where $h(X, L)$ is the measurement prediction function, and Σ is the measurement covariance matrix.

This naive version fails to minimize computational cost because each robot solves

[2] *The DDF module was developed by Alex Cunningham*

the same large optimization problem, and sends a large volume of information over communication channels. We address the failings of this system through DDF-SAM in two ways: 1) each robot optimizes its own local map before sharing with other robots and 2) we only share compressed versions of local maps. The DDF-SAM system consist of three modules: the *local mapping* module, which performs local nonlinear optimization and map compression, the *communication* module, which shares compressed maps with other robots and maintains a cache of known maps, and a *global mapping* module to assemble the compressed maps.

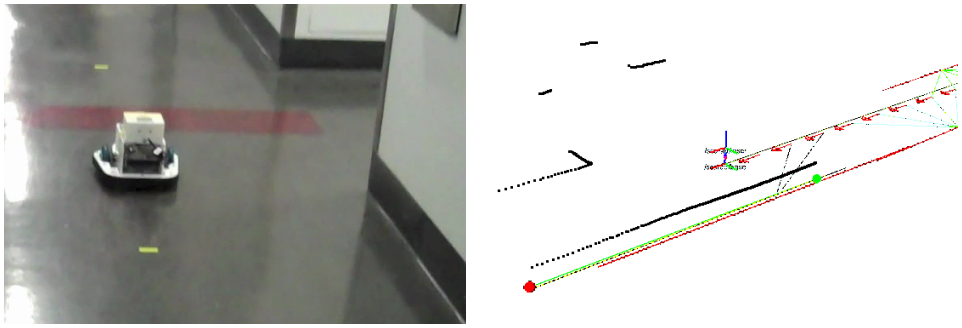
At the local mapping level, the problem is identical to single robot mapping, with an additional step that extracts the landmarks from the map by marginalizing out robot poses to form the compressed map. A compressed map M^r consists of a set of landmarks L^r in the reference frame of robot r, as well as a linear system in square-root information form (A, b). The size of these compressed maps only grows with exploration, which minimizes communication size.

The communication module both broadcasts and receives these compressed maps from any neighboring robots, and caches the latest compressed map for each. This module also assembles the global factor graph combining compressed maps from all of the robots. Due to the linearization of the compressed maps and unknown global reference frames, the global system incorporates a set of transforms T^r for each of the robots that transform global landmarks l_i into local landmarks l_i^r.

To encode the relationship between the landmarks in the local frame of a compressed map and a globally consistent set of landmarks, we add hard equality constraints to the factor graph expressing the relationship $T^r \oplus l_i = l_i^r$. We implement these nonlinear constraints as penalty functions, and the global optimization module solves simultaneously for the relative reference frames, a global copy of landmarks, and updates to the compressed maps. Because each robot performed local optimization before sharing its map, the computational cost of fusing compressed map is

negligible.

3.2.3 EVALUATION



(a) Situation where another robot is recruited

(b) Map acquired by robot 1

Figure 21: Situation where another robot is recruited for exploration of the second hallway

The designed system was tested across more than 10 runs in the environment shown in figure 18. The mission is launched at the lower right hand corner of the environment. As the first robot passes the initial intersection, it recruits another robot (using the mesh network). The second robot drives to the intersection using localization and the partial map provided by first robot. The situation is illustrated in Figure 21(a) and the corresponding map is shown in Figure 21(b). As the second robot turns right and runs down the long hallway to the right it will encounter the middle part of the figure-eight shaped environment and recruit a third robot. As the team explores the full environment, a complete map of the environment is generated, as shown in Figure 22

In the environment presented here, only three robots were utilized. In general, the number of branch points in a graph will represent the number of "assist" requests generated. As robots complete their mapping missions (reaches end of a hallway or returns to a location previously mapped), they are available to respond to such requests.

The distributed mapping system runs in real-time on the 1.5 GHz processors on

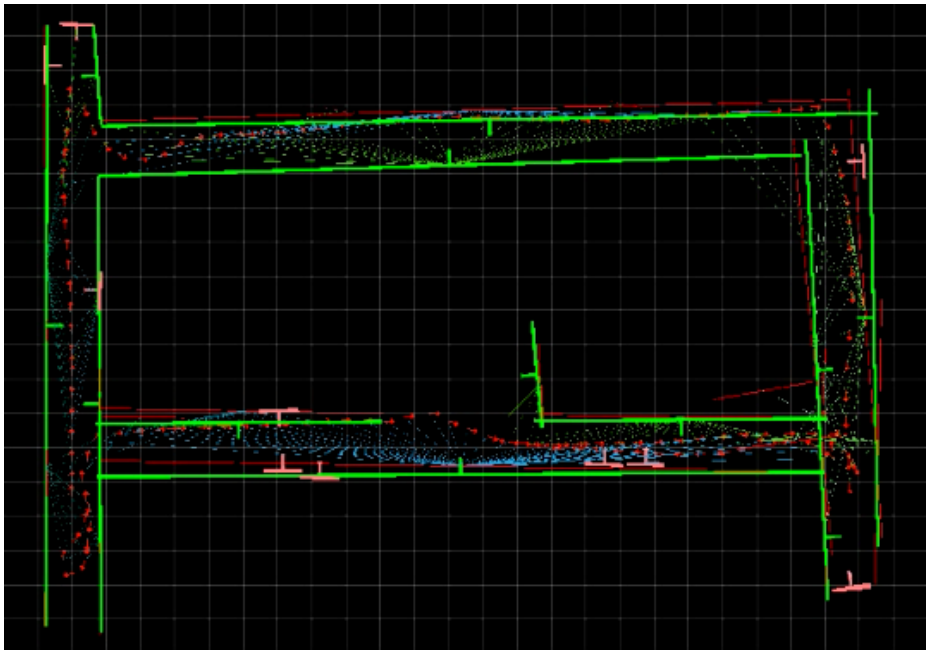


Figure 22: The map generated for the complete environment

the Scarabs. The system generates a map that is accurate to within 5-10 cm across the environment which is 15x25m in size.

A challenge in the present system is that data association must be close to perfect, as the system is challenged in recovering from incorrect updates of the map. At the time of these experiments, we had not implemented probabilistic data association techniques; however, we have now implemented the joint compatibility technique from Neira and Tardós [2001].

3.2.4 Summary of autonomous 2D multi-robot mapping

There is a tremendous potential to utilize robot systems for cooperative exploration of unknown environments. As the size of robots is reduced to be almost palm sized they can be thrown into a building through a window or easily pushed through a small door opening. In such situations, it is of interest to have methods that enable automatic exploration of an environment with load-sharing to ensure efficient coverage of

the full environment. We have presented an example system that enables fully autonomous exploration of indoor environments. A reference implementation has been provided and it has been tested across environments at Georgia Tech and University of Pennsylvania (only the UPENN results were presented here). The system does real-time exploration and provides maps of the environment in terms of doorways and walls. Both of these features represent important structures to be considered by first responders or soldiers as they enter an unknown building.

In this section, we presented a strategy for multi-robot mapping in relatively sparse settings. In environments with richer structure and more complex layouts, there is a need to consider the use of more complex features. A technique for incorporating 3D feature measurements with a heterogenous team of mobile robots will be presented in the next section.

3.3 Cooperative 3D and 2D Mapping With Heterogenous Ground Robots[3]

Unmanned vehicle systems (UVS) can be used to provide situational awareness to first responders. Current 2D mapping technology, as was introduced in section 3.2, is unable to adequately represent elevation transitions in multi-story structures or represent detail that exists outside of a single plane in the environment. Recent advancements in 3D sensing technologies enable algorithms for teams of autonomous agents to build 3D maps of the environment. These 3D maps can be used by first responders or soldiers to plan and coordinate ground operations with rich information while minimizing risk to personnel.

The mapping system presented in this section makes 3D measurements of features in the environment. These measurements are optimized in a graph to determine the structure of the environment and the trajectory of the robot. Full 3D maps are

[3] ***This section is based upon Rogers III et al. [2012a]***

generated by rendering sensor data along the robot's optimized trajectory, which can be used for operational planning. 3D feature mapping is desirable because of the accuracy and completeness of rendering map information which can be interpreted by users. Unfortunately, robots which can acquire 3D maps are more complex and expensive than robots which can acquire 2D maps. In addition, 2D data can be more efficiently analyzed and can produce measurements in real-time; 3D data can only be analyzed for map measurements at a much slower rate. In this section, we consider an alternative heterogenous team consisting of a robot with a 3D sensor in addition to a robot with a 2D sensor which work together to build a complete map of an indoor environment.

These algorithms have been implemented on a team of iRobot PackBot systems that have been augmented with an onboard computer. One of the robots, referred to as the *3D robot* in this section, has a 3D laser scanner. The other robot, referred to as the *2D robot* in this section, has a 2D laser scanner.

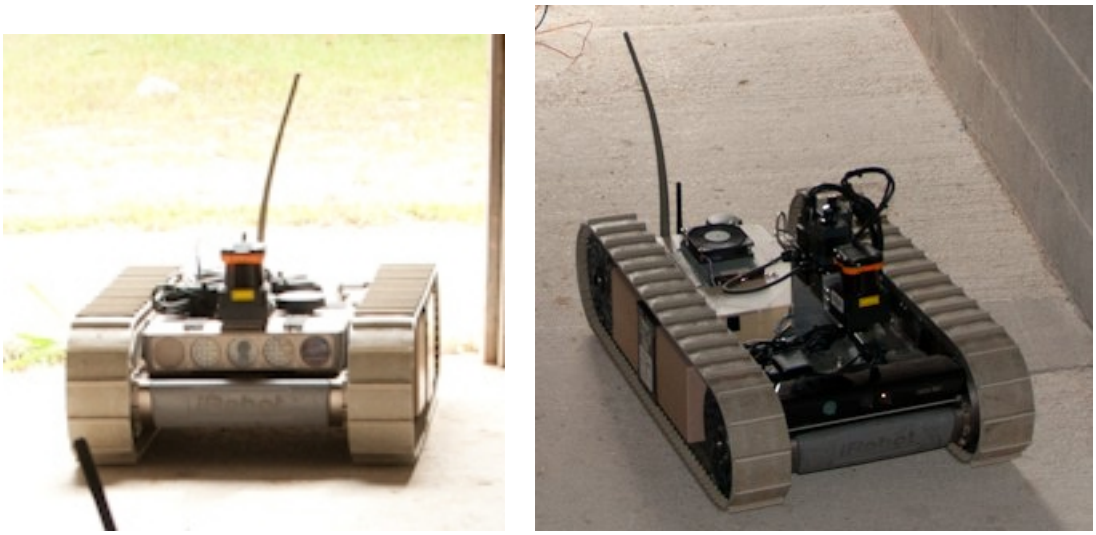
Experimental evaluation of mapping performed by this robot team has been conducted at a US military MOUT site which simulates the challenges in modern urban warfare. The robots are tele-operated in one of the structures at the MOUT site. The resulting maps generated by each robot alone are compared to a map generated by both robots working together.

To integrate measurements from multiple robots into a single map, we make use of a version of OmniMapper which maintains multiple robot trajectories with shared landmarks. Each individual robot builds a local map and sends relevant map data to the central map server. On the central map server, measurements are data associated to existing or new landmark entities via an implementation of joint-compatibility branch and bound (JCBB) [Neira and Tardós, 2001].

Sensor data is analyzed to produce feature measurements through the line extraction module and the plane extraction module. The line extraction module is a

RANSAC-based technique described in Nguyen et al. [2005] and detailed previously in section 2.4.2. The plane extraction module is also a RANSAC-based technique from Rusu and Cousins [2011] which was detailed previously in section 2.4.4. As the number of points evaluated by the plane extraction module is many orders of magnitude greater than the number of points evaluated by the line extraction module, plane extraction requires several seconds per frame while line extraction occurs in real-time.

3.3.1 Experiments



(a) An iRobot PackBot equipped with a 2D laser scanner.

(b) An iRobot PackBot equipped with a pan-tilt unit and a laser scanner, capable of making 3D measurements of planar features such as walls.

Figure 23: The team of iRobot PackBots which are used in the experiments in this section

Experiments were performed with two iRobot PackBots as seen in figure 3.3.1. One robot is performing 2D line mapping of walls with a Hokuyo UTM30 laser scanner in figure 23(a). The second robot is equipped with a Hokuyo UTM30 laser scanner which is mounted on a Directed Perception PTU-46-70 pan-tilt unit in figure 23(b). This pan-tilt unit is actuated and synchronized with the laser scanner to produce a 3D point cloud of the walls in the environment.

(a) An indoor setting at the MOUT site being mapped by an iRobot PackBot.

(b) One of the rooms in the *Hotel*

Figure 24: Example scenes from the MOUT training site where these experiments were performed.

The experiments are performed on log data taken at the MOUT (military operations in urban terrain) training/test site at Camp Lejeune US Marine Corps base in North Carolina, shown in figure 3.3.1. The buildings where the experimental data was collected are designed to simulate typical urban combat terrain. The building where the data which is used in this section was collected is called the *Hotel*; it is typical of a hotel with many smaller rooms with adjoining bathroom-sized rooms connected with main hallways.

When the data was collected for the experiments in this section, we tele-operated the robots. We currently are experimenting with autonomous collaborative exploration and mapping; however, this autonomous control was not available when these experiments were carried out. Under tele-operative control, we carefully selected the paths of the robots to perform three classes of collaborative mapping.

In the first class of collaborative mapping experiments, the two robots follow the same path. Both robots start in the second-floor atrium and immediately turn to their right and move through a room and out into a large room and back into the atrium. Here the loop is closed back to where the robots started. The 3D mapping robot proceeds in front of the 2D robot, which follows about 3 meters behind. Obviously, here the two robots are unable to achieve any time efficiency benefit when both are used because they are redundant; however, we can expect that the thoroughness of their mapping should feature additional detail that neither robot could have achieved alone.

In the second class of collaborative mapping experiments, there is very little overlap in robot trajectories. Both robots start in the second-floor atrium and move into the main hallway. The 3D mapping robot turns left; the 2D mapping robot turns right. Several rooms along the hallway are explored by each robot. The robots then proceed back to the start locations. Here we can expect that the time efficiency of the mapping operation is doubled from what a single robot could have accomplished (or

the space covered is doubled in the same amount of time), while each robot might miss some detail and cannot leverage the efforts of their counterpart to improve accuracy.

The final class of collaborative mapping experiments is a hybrid of the first two classes. Here, the robots will explore rooms off of the same branch of the hallway. The robots overlap a significant portion of their trajectories, and explore separate alternating sets of rooms along the hallway. Here we expect to see good time efficiency improvement over single robot mapping, and the robots should also improve accuracy as more observations are made with each sensory modality.

3.3.2 Results

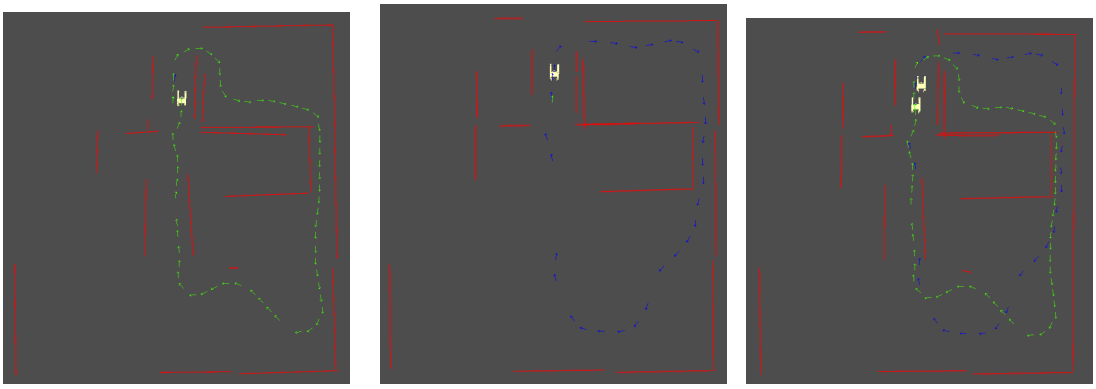
We collected five test data sets and present the results from three of them here which are representative of the three test cases detailed in the previous section: fully overlapping, mostly non-overlapping, and partially overlapping.

The results from the fully overlapping test run can be seen in figure 25. The 2D robot follows several meters behind the 3D robot. Both robots perform their respective mapping tasks well; however, some detail is lost in the 3D robot only map due to the low frequency nature of this measurement modality. Since the loop is closed by all of these maps successfully, the use of multiple robots does not significantly impact accuracy; however, the rightmost wall in the combined map is fused into a single wall, whereas it appears as broken segments in both individual maps. This complete wall is an important landmark because it is rigid and more useful for maintaining map correctness.

The results from the mostly non-overlapping test run can be seen in figure 26. The 3D robot proceeds down the left branch of the main hallway; the 2D robot proceeds down the right branch. The only overlapping portion of the two maps is in the starting location. Both robots map their respective wings of the *Hotel* and

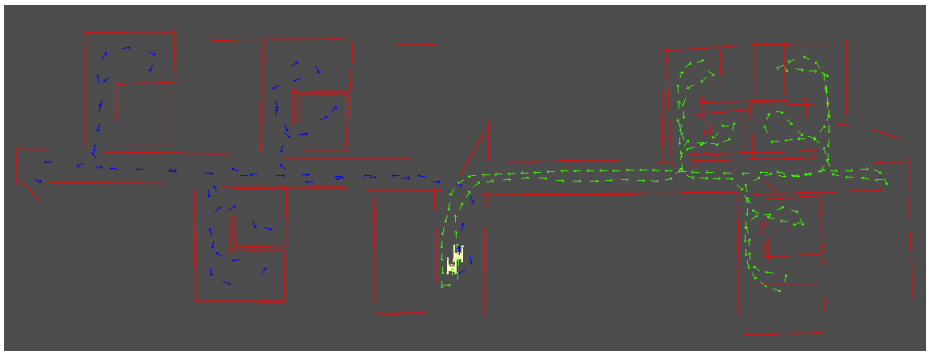
return to the start location. We claim that this experiment indicates that multi-robot mapping with heterogenous sensor modalities can improve time efficiency for the mapping task. Accuracy is not noticeably improved since both individual maps were generated correctly in this case.

The results from the partially overlapping test run can be seen in figure 27. In this test run, the two robots proceed down the same hallway, but they alternate rooms for mapping. Here, the 2D mapping robot makes some mistakes due to calibration and data association errors and an additional wall is placed in the environment along the main hall. The 3D mapping robot makes no such mistakes. When their map data is fused, the 2D mapping robot is able to map more correctly. Since both robots were operating simultaneously and did not interfere with each other, we contend that this indicates that this system is capable of building a map more efficiently while increasing accuracy.

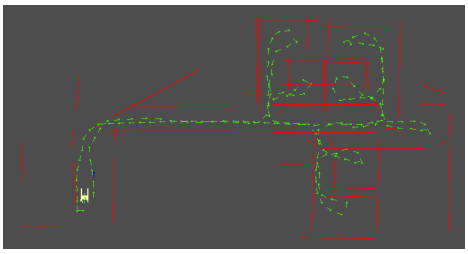


(a) 2D robot only (b) 3D robot only (c) Both 2D and 3D robots

Figure 25: Maps generated from data collected in the first class of test run. Both robots follow the same trajectory through a room adjacent to the starting area in the second floor atrium. The 2D mapping robot follows several meters behind the 3D mapping robot.



(a) Both 2D and 3D robots



(b) 2D robot only

(c) 3D robot only

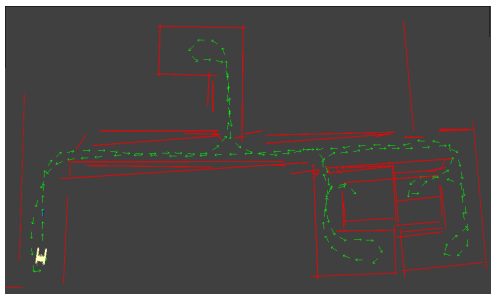
Figure 26: Maps generated with data collected in the second class of test run, with minimal overlap where both robots move in opposite directions to explore the entire length of the main hallway.

3.4 Collaboration strategies for multi-robot exploration and mapping[4]

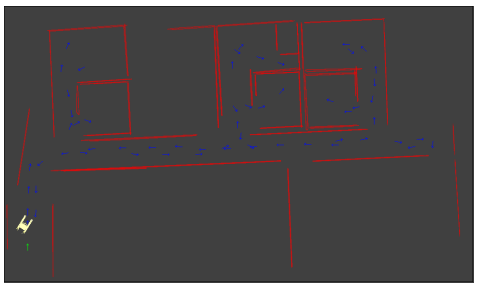
Projects like the Army Research Laboratory's Micro-Autonomous Systems Technology (MAST) [ARL, 2006] seek to introduce the application of large numbers of inexpensive and simple mobile robots for situational awareness in urban military and rescue operations. Human operators are required to teleoperate the current generation of mobile robots for this application; however, teleoperation is increasingly difficult as the number of robots is expanded. There is evidence in human factors research which indicates that the cognitive load on a human operator is significantly increased when they are asked to teleoperate more than one robot [Zheng et al., 2011].

Autonomy will make it possible to manage larger numbers of small robots for

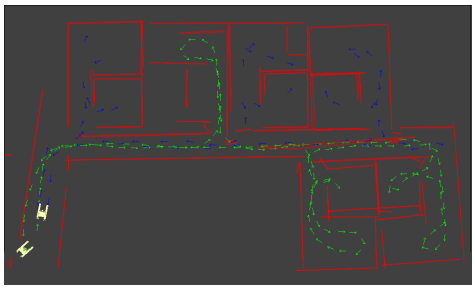
[4] ***This section is based upon Rogers III et al. [2012b], and new experiments in a journal version (IJRR) which is under review***



(a) 2D robot only



(b) 3D robot only



(c) Both 2D and 3D robots

Figure 27: Maps generated from the third class of multi-robot mapping, one in which there is partial overlap.

mapping. There is a continuum of options as to the degree of shared autonomy between robot and human operator [Heger and Singh, 2006]. Current robots employed in explosive ordinance disposal (EOD) missions are fully tele-operated. At the other extreme, robots can be given high-level tasks by the operator, while autonomously handling low-level tasks [Chipalkatty et al., 2011] such as obstacle avoidance or balance maintenance. In this section, our robot teams occupy the latter end of the spectrum; we imagine that the operator has tasked the robot team to autonomously explore and map an unknown environment while focusing on the high level task of looking for survivors.

In the multi-robot scenario, resources are distributed amongst a team of robots instead of concentrated on one large and expensive machine. This distribution offers a number of advantages and disadvantages over the single robot case. The distributed team is able to continue its mission even if some of the robots are disabled or destroyed. A single robot can only explore or monitor at one location at a time; however, the

multi-robot team can provide situational awareness in many locations at once. Unless the single robot is able to move much faster than the multi-robot agents, the lone robot will be slower in performing the exploration and mapping task. These advantages are taken for a multi-robot team at the cost of increased complexity in communication and coordination.

As the number of robots is increased, each robot may interfere with one another and eventually decrease the performance of the mapping task. Careful consideration of exploration strategy and coordination of large numbers of mobile robots can efficiently allocate resources to perform the mapping task more quickly and more accurately.

Multi-robot mapping and exploration was addressed in Fox et al. [2006] and Vincent et al. [2008]. These papers build a map using up to 3 robots with a decision-theoretic planner which trades off robot rendezvous operations with frontier exploration. These robots rendezvous to determine their relative pose transforms to provide constraints to recover the final map. In contrast, our approach does not require this rendezvous step because landmarks are globally data associated between each robot on a central map coordinator. The exploration strategy used is similar to our strategy called *Reserve*; however, we will not use a rendezvous step and do not require a decision-theoretic planner.

3.4.1 Technical Approach

3.4.1.1 Mapping System

We use the library called *OmniMapper* described in section 2.4. In this application, the mapper is using the *plane* mapping plugin described in section 2.4.4.

Each robot in the team builds a map locally with the *OmniMapper* and sends map data to the map coordinator. Each robot can incorporate new landmark measurements whenever it has moved far enough from the last pose where measurements were made. In the current implementation this is set to 10cm. When a robot finishes

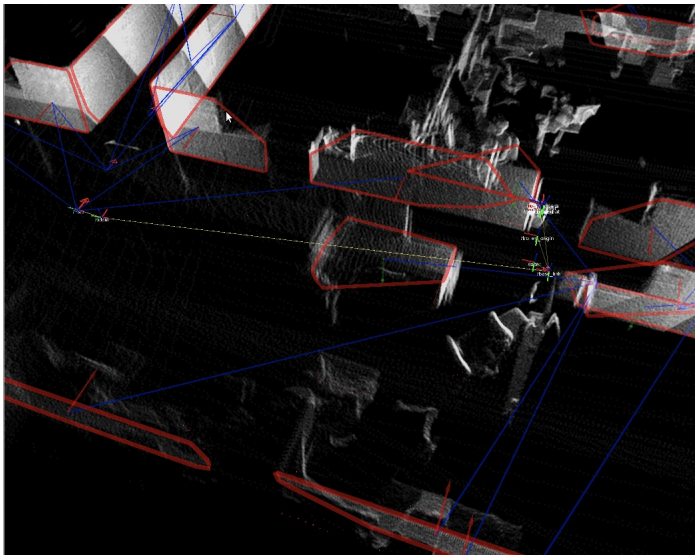


Figure 28: OmniMapper.

optimizing its local map with new landmark measurements, all relevant information needed by the map coordinator is packaged and transmitted.

The information which is needed by the map coordinator to incorporate a new piece of information from a team member consists of many components. First, the sensor measurement data is needed. In the current implementation, this consists of the extracted plane information consisting of a plane equation along with a convex hull of points along the perimeter of the plane. This represents a significant compression over an alternative scheme where all point-cloud data could be transmitted and processed at the master node. Secondly, the team member's integrated odometry is transmitted. This allows the master node to compute the odometric relative pose since the prior landmark measurement data was incorporated; this is used to insert a relative pose factor and also give initial conditions for data association. Finally, the team member's local map pose is transmitted. This is used by the master node to compute a map pose correction. This correction is sent back to the team member so that it knows it's relative pose in the global map frame. This knowledge is needed so that the team member can interpret exploration goals correctly.

The map coordinator maintains trajectories for each of the robots in the team. Measurements from each robot are merged into one global view of the landmarks. This is realized through a simple modification to the standard *OmniMapper* through duplication of data structures tracking indexing data and pose information used for interaction with GTsam into arrays. This implementation potentially allows for an unlimited number of team members to build a map together.

Most modern SLAM approaches use a *pose graph* [Grisetti et al., 2007] which is generated via laser scan matching in 2D or point-cloud ICP in 3D. This approach is effective for single robot mapping; however, it has some drawbacks for larger multi-robot mapping. Scan matching and ICP algorithms are computationally intensive and matching across many robots would rapidly become intractable. Also, point cloud representations are large and their transport over a wireless link could be prohibitive if the link is limited in capacity due to mesh network routing or environmental interference. To address these limitations, our robots extract relevant, parsimonious features from the environment and transmit them to the master node.

Each turtlebot in these experiments maps planar wall structures using a Microsoft Kinect sensor. Planar segments corresponding to walls are extracted from point clouds via a RANSAC [Fischler and Bolles, 1981] based algorithm [Rusu and Cousins, 2011]. This process was described in section 2.4.4.

The Kinect sensor on each robot has a narrow field-of-view which is not ideal for detecting exploration frontiers. To alleviate this problem, we incorporated a strategy by which each robot will rotate periodically to get a 360 degree view of its surroundings. This data is synchronized with robot odometry to synthesize a 360 degree laser scan. This synthesized laser scan is sent to the local mapper and forwarded to the global mapper. At the global mapper, it is linked to a trajectory pose element and used to populate an occupancy grid. This occupancy grid is re-computed after every map optimization so that a loop closure will result in a correct occupancy grid map.

The frontier based exploration strategies detailed below use this occupancy grid to find the boundary between clear and unknown grid cells.

3.4.1.2 Exploration Strategy

Each robot team leader uses a frontier based exploration strategy similar to the one used in Vincent et al. [2008]. An exploration frontier is defined on a costmap cellular decomposition where each cell has one of three labels: *Clear*, *Obstacle*, and *Unknown*. The costmap is initialized as *Unknown*. Costmap cells are set to *Obstacle* corresponding to locations where the Kinect sensor detects an obstacle in the environment. The cells on a line between the obstacle cell and the robot's current location are set to *Clear*. Exploration frontiers are defined as *Clear* cells which are adjacent to at least one neighbor where the label is *Unknown*.

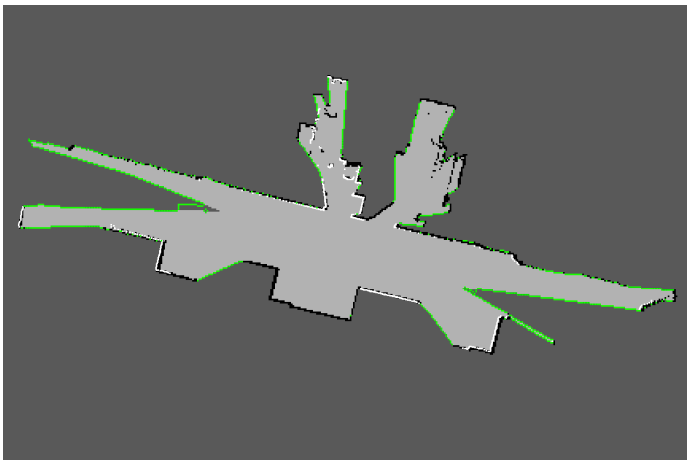


Figure 29: Global maps using the *Reserve* coordination algorithm described in this section.

The high level robot exploration goal allocation is centrally planned on the same workstation where the global map is constructed. There are many choices which can be made by the exploration planner when choosing which robot or group of robots should move towards an exploration goal. We have chosen to employ a greedy strategy by which the nearest robot or team is allocated to a goal instead of a more sophisticated traveling-salesman type of algorithm. We believe that this is appropriate

because the exploration goals will change as the robots move through the environment; re-planning will be required after each robot or team reaches an exploration goal.

3.4.1.3 Coordination Strategy

The coordination strategy used between robot agents as well as the number of robots are the independent variables in the experiments performed in this section. The coordination strategy refers to the proportion of robots which are dispatched to each exploration goal. On one extreme, a single robot can be sent to explore a new goal; at the other extreme all available robots can be sent to a new goal. Larger robot teams sent to a new exploration goal will improve availability of new agents at the location of new exploration goals are discovered. The larger group has spare robots which can be quickly allocated to explore new goals, such as those discovered when the team moves past a corridor intersection or t-junction. If the group of robots allocated to a navigation goal is too large, then the robots can interfere with each other due to local reactive control of multiple agents with respect to dynamic obstacles and limited space in corridors. The strategies selected for testing trade off *availability* (robots are close and able to explore branching structure quickly) with *non-interference* (robots do not get in each other's way).

The first coordination algorithm is called *Reserve*. In this algorithm, all unallocated robots remain a the starting locations until new exploration goals are uncovered. When a branching point is detected by an active robot, the closest reserve robot will be recruited into active status to explore the other path. This strategy has low availability because all of the reserve robots remain far away at the entrance; however, it has minimal interference because the exploring robots will usually be further away from other robots.

The second coordination algorithm is *Divide and Conquer*. In this strategy, the entire robot group follows the leader until a branching point is detected. The group

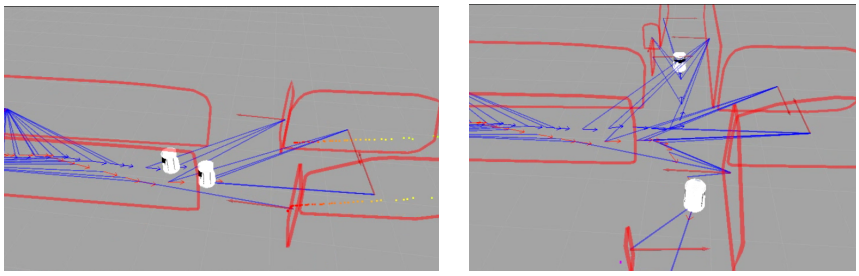
splits in half, with the first $\frac{n}{2}$ robots following the original leader, robot $\frac{n}{2}+1$ is selected as the leader of the second group, and robots $\frac{n}{2}+2$ through n are now members of its squad. Once there are n squads with one robot, no further divide operations can be made and new exploration goals will only be allocated once a robot has reached a dead-end or looped back into a previously explored area. This algorithm maximizes availability, but potentially causes significant interference between robots.

The third coordination algorithm is called the *Buddy System*. In this strategy, robots are recruited from the reserve pool in teams of two. When a branching point is detected by a full team of two robots, the team will split into two and proceed along both paths. When these single robots detect additional split points, new teams of two robots will be allocated out of the reserve pool and they will explore this new goal and divide when another branching point is reached. This strategy uses small teams of robots which are able to maneuver around one another without too much interference, while maintaining good availability to respond quickly to explore new frontiers.

An example 3D map built by two robots as they approach a branch point can be seen in figure 30(a). At this point, the robot team splits and each team member takes a separate path, as seen in figure 30(b). The map shown is built concurrently with local maps built on each robot. The global map is used to establish a global frame of reference for robot collaboration message coordinates.

3.4.2 Experiments

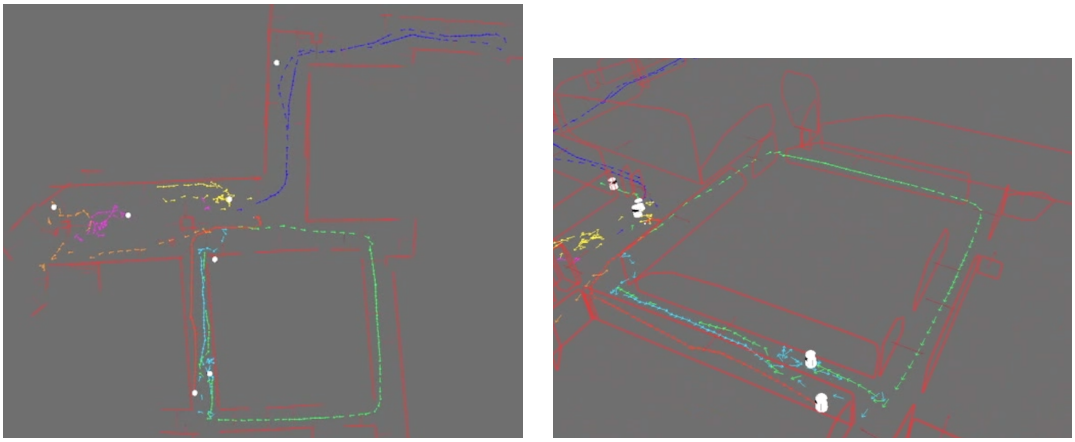
The setting for the multi-robot mapping task for this series of experiments consists of a team of robots being introduced into a single entrance in an unknown environment. Each robot is an inexpensive Willow Garage *TurtleBot*; a team of nine of these robots is shown in figure 32. The *TurtleBot* was chosen for this application due to its low cost and the ease of integrating large numbers of robots through ROS. The *TurtleBot*



(a) Two robots approach the intersection.

(b) Two robots split and move past the intersection

Figure 30: An illustration of the *Divide and Conquer* exploration strategy. As the robots approach an intersection, the team must split and recruit new partner robots from the reserved units.



(a) A map built by seven robots in an experiment using the *Reserve* cooperative mapping strategy.

(b) The same map shown from a different angle to demonstrate 3D plane features which are used for map landmarks.

Figure 31: Global maps gathered by a team of seven mobile robots.

platform is based on the iRobot *Create* base. The robots make measurements of planes with a Kinect sensor, and use an onboard IMU together with odometry to estimate ego-motion.

We evaluated the performance of various robot coordination strategies in the multi-robot exploration and mapping task. An example scenario for the *Divide and Conquer* cooperative mapping strategy can be seen in the panorama image in figure 33.

In the first series of live robot experiments, we evaluated the first two strategies

Figure 32: Our nine TurtleBots used in these experiments.

Figure 33: An example scenario for the experiments described in this section. Three teams of two robots are exploring the branching hallway structure in an office environment. In this illustration, the robots are using the *Divide and Conquer* cooperative mapping strategy.

which were developed, the *Reserves* and *Divide and Conquer*. This series of experiments was designed to demonstrate the performance of these two cooperative exploration strategies. A total of 6 runs were performed for each cooperation strategy, team size, and starting location. For each experiment run, the *TurtleBot* team explored the environment from a wedge-shaped starting configuration, which can be seen in figure 32. These experiments were performed in an office environment. In order to measure the exploration and mapping performance in each location, we chose specific starting locations which are labeled *Base1* and *Base2* in figure 34. These starting locations were chosen because the area around the robot teams could be blocked off so there is only one initial exploration frontier, directly in front of the lead robot. This initial configuration was chosen to represent a breaching behavior which would

be needed for implementation of collaborative mapping in a hostile environment.

Figure 34: The office environment where the experiments were performed. The areas labeled Base1 and Base2 are the initial position of the robots. Red lines indicate artificial barricades to restrict the initial exploration of the robot teams to simulate a breach entrance into a hostile environment. Blue squares indicate the position of points-of-interest. Results are reported on the number of these points-of-interest visited by the robot team.

In the second series of live robot experiments, we evaluated all three collaboration strategies *Reserves*, *Divide and Conquer*, and the new strategy *Buddy System* in various buildings in a training facility designed to simulate an urban environment. Due to the remoteness of this training facility we only brought five robots for testing. The robot team size is varied from three to five robots. We attempted to run three test runs are made for each combination of team size and strategy across three buildings for a total of 27 experiments.

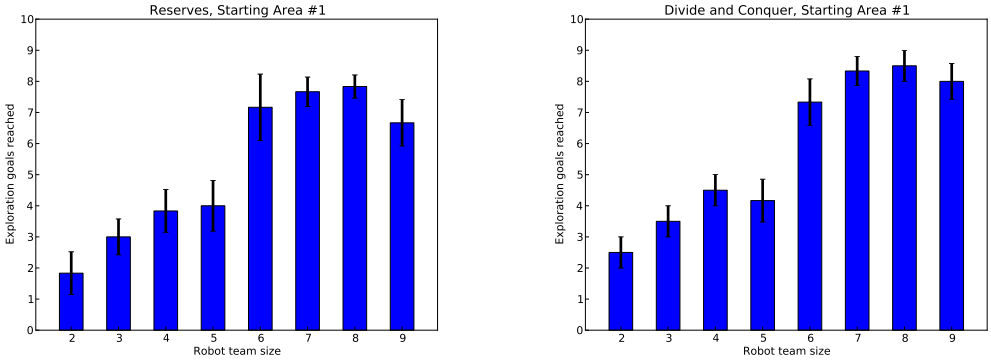
3.4.3 Results

The first series of experiments demonstrate team performance based upon coverage in a mapping task on an unknown office environment. Robot team sizes were varied from 2 to 9 robots. An map built with 7 robots at TurtleBots using the *Reserve* strategy is seen in figure 31(a). An image showing the same final global map from a side view demonstrates the 3D plane features in figure 31.

Each of the collaboration strategy and robot team size experiments were performed from two starting locations. These starting locations are labeled *Base1* and *Base2* in figure 34. A series of interesting locations was determined in advance by examining the building floor-plan; these points of interest are also marked in figure 34. Each experiment run gets a score based on how many of these points of interest are visited and mapped before a time limit is reached. This score represents the effectiveness of that algorithm and team size at providing coverage while exploring an unknown map.

In the first experiment series from *Base1* in figure 34, both strategies achieve reduced exploration coverage per robot as the team size is increased, as can be seen in the graphs in figure 35. In this starting location, there is limited space to maneuver, so both strategies generate significant interference between robots trying to move to their goals. In several instances, pairs of robots even crashed into each other due to the limited field-of-view of their sensors. We believe that the *Divide and Conquer* strategy results in figure 35(b) indicate that the team was slightly more effective than the *Reserves* strategy in figure 35(a). At the largest team size of 9 robots, the *Divide and Conquer* strategy usually visited one additional point-of-interest more than the *Reserves* strategy. Additional qualitative impressions are that the *Divide and Conquer* strategy explored the points-of-interest that it reached more quickly than with the *Reserves* strategy. For both strategies, the best team size appears to be 6 robots in this starting location.

In the second set of the first series of experiments, the robot teams were placed



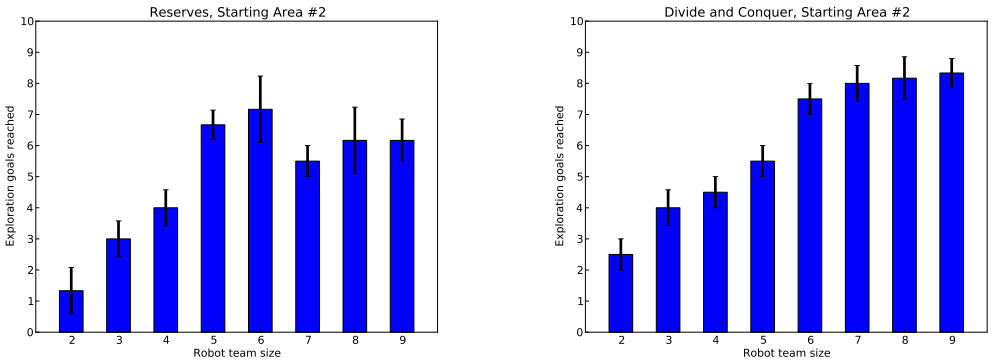
(a) Reserves

(b) Divide and Conquer

Figure 35: Results from the first starting area

in the starting area labeled *Base2* in figure 34. As in the first experiment, the per-robot performance of both strategies decreased as the number of robots were increased. This series of experiments demonstrates a marked improvement of the *Divide and Conquer* strategy over the *Reserves* strategy as can be seen in figure 36. The *Divide and Conquer* strategy causes more robots to be making observations of exploration frontiers due to the fact that groups contain more than one robot. These additional observations of the frontier allow the *Divide and Conquer* strategy to find exploration frontiers faster than the *Reserves* strategy, and therefore explore more points-of-interest. The second experiment started from an area where there is more room to maneuver. This allowed the *Divide and Conquer* strategy to have less interference since the entire team moved together out of the starting area into the larger area before any divide operations were performed. The *Reserves* strategy still had to initially maneuver from the cramped starting location. As in the first experiment, the *Divide and Conquer* strategy qualitatively explored the environment faster than the *Reserves* strategy. The best value for the number of robots is 6, which is the same value found in the first experiment.

In the second series of experiments we ran three trials of each strategy in three different buildings shown in figures 37(a), 37(b), and 37(c). Each trial lasts up



(a) Reserves (b) Divide and Conquer

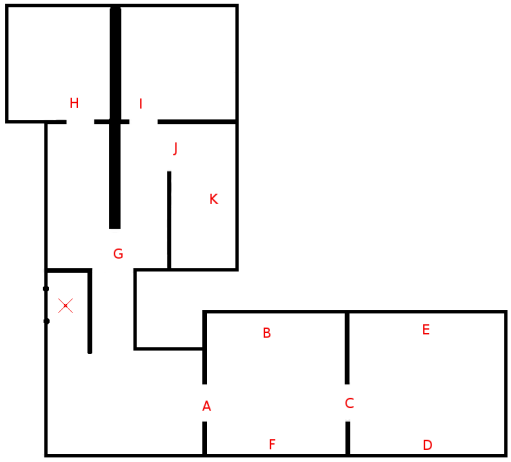
Figure 36: Results from the second starting area

to 36 minutes before the teams have finished exploring the complete floor of the building. The set of locations were manually labeled by room separation. Some of these locations are difficult to reach given the physical capabilities of the robots. In figure 38(a), the robot needs to navigate through a narrow hallway in order to explore the rest of the floor of building C (37(c). For example in figure 37(b), the robot needs to reach the goal labeled P and O in order to complete the exploration of the room.

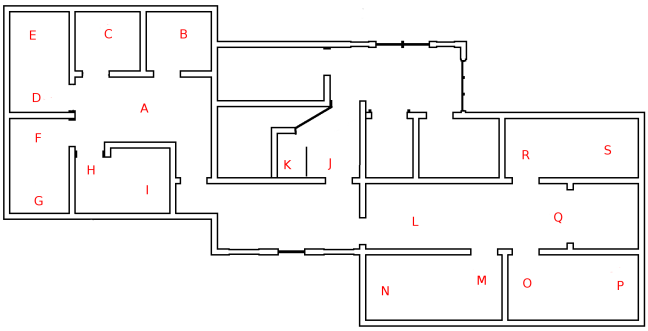
These experiments were performed with teams of three, four and five robots. Figure 38(c) shows the initial configuration of one of the experiments ran in Building C. This initial configuration needs to be set in order to fit the robots close to a door. This is meant to simulate an exploration task for a rescue mission which starts by introducing the robots through a doorway into the building. As shown in figure 38(b), the robots explore and navigate in an environment which exhibits difficult lighting conditions; however, this is not a problem for the Microsoft Kinect sensor.

3.4.4 Overall performance

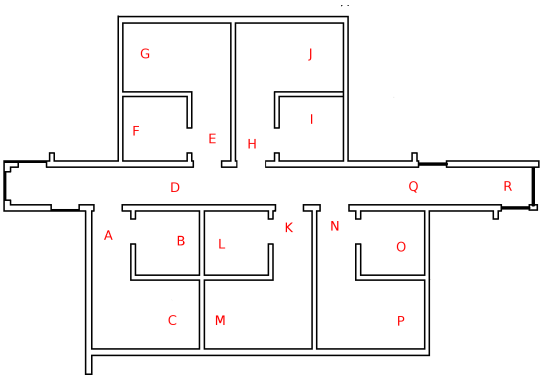
Our results are summarized in tables 1, 2, 3 show the average on time that each group of robot takes to explore the three buildings. In comparison with the simulation experiments, the implementation in real robots are subject to hardware failures and a hostile environment with debris and small potholes. In some of the experiments,



(a) Building *A*.



(b) Building *B*.



(c) Building *C*.

Figure 37: Architectural floor plans for buildings used for robot exploration and mapping experiments. Navigation key points used for scoring runs are indicated by letters on the maps.

(a) Robots navigating through narrow hallways in building B

(b) Three robots exploring Building B

(c) The initial configuration of the robots in Building A, intended to simulate a breach entry from the door.

Figure 38: Robot teams running exploration and mapping experiments

Table 1: Average time of each exploration strategy with 3 robots. Buddy System was not run in Building C. Statistics are gathered from three runs.

Strategy	Building A	Building B	Building C
Divide and Conquer	2233 ± 56	1718 ± 287	1127 ± 30
Reserves	$2526 \pm -$	1699 ± 58	1521 ± 241
Buddy System	1989 ± 58	1531 ± 86	$-$

particularly those which took place in building C, this caused a high degree of variability between runs. However, running the experiments in the other two environments provided a good estimation of the performance of each exploration strategy.

We analyze each strategy in terms of the average time that the robots took to finish the exploration task given team size and collaboration strategy. As the results indicated in figures 39(a), 39(b), and 39(c), the strategies *Divide and Conquer* and

Table 2: Average time of each exploration strategy with 4 robots. Buddy System was not run on Building C. All other statistics are gathered from three runs, except Reserves in Building C was only run once.

Strategy	Building A	Building B	Building C
Divide and Conquer	1997 ± 46	1188 ± 93	773 ± 31
Reserves	2221 ± 95	1271 ± 36	$1602 \pm -$
Buddy System	1991 ± 182	1237 ± 90	$-$

Table 3: Average time of each exploration strategy with 5 robots

Strategy	Building A	Building B	Building C
Divide and Conquer	1437 ± 61	689 ± 35	920 ± 69
Reserves	1832 ± 38	947 ± 83	$1689 \pm -$
Buddy System	1529 ± 212	685 ± 66	$-$

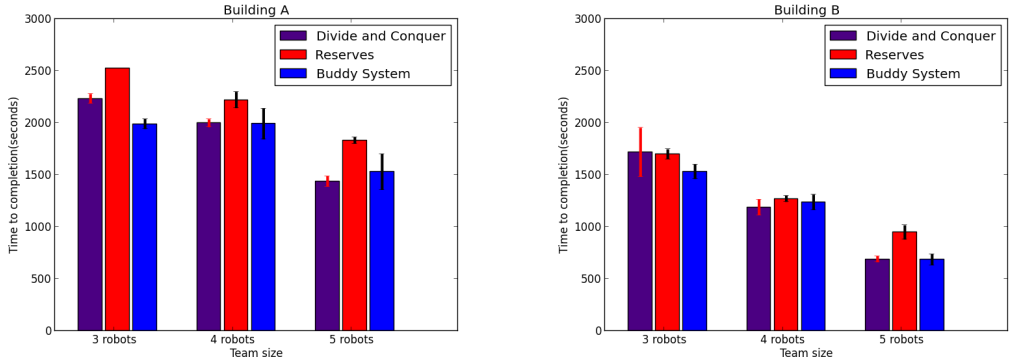
Buddy System always performed better than the *Reserves* strategy. The performance also always increases as the team size is increased in buildings A and B. This results is consistent with the office building results presented in the first series of experiments. Those experiments showed decreased performance improvement only once the team size was increased beyond 6 robots, which is larger than the teams we were able to test in this experiment series. The team performance is not improved in building C due to the poor conditions in that building including debris and concrete seams which inhibited progress.

3.4.5 Discussion of collaboration strategies for multi-robot mapping

We have presented experiments which evaluate three collaboration strategies which can be used by teams of mobile robots to map and explore an unknown environment. We have also evaluated the impact of the number of robots on coverage in the exploration and mapping task.

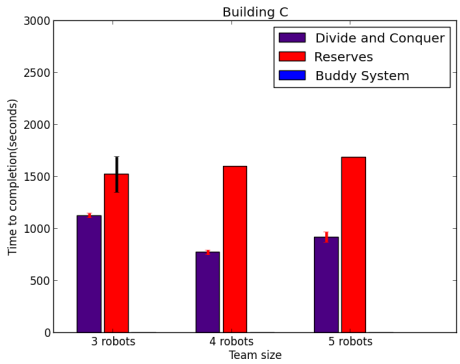
The first collaboration strategy, called *Reserves* keeps a pool of unallocated robots at the starting location. A new robot is activated when there are more exploration frontiers than currently active robots. This strategy was intended to minimize the amount of interference between robot agents since robots would be far away from each other during exploration. The results from our experiments do not indicate that this strategy results in less interference than other strategies since performance decreases more when more robots are added in some environments. The *Reserves* strategy is significantly slower at exploring the environment than other strategies.

The second collaboration strategy, called *Divide and Conquer* has all available



(a) Building A

(b) Building B



(c) Building C

Figure 39: Comparison of the strategies and team sizes the buildings at the test facility. Note that the *Buddy System* was not tested in Building C due to time constraints.

robots proceed in one large group. Once there are two exploration frontiers, at a corridor T-junction for example, the team will divide in half and each sub-team will follow one of the exploration frontiers. This process will be repeated with teams dividing in half each time they see branching structure in the environment. It was anticipated that this strategy would result in higher interference since robots would be maneuvering close together; however, the increased availability of robots near new exploration frontiers offsets this phenomenon.

The third collaboration strategy, called *Buddy System*, gives some of the advantages of the *Divide and Conquer* strategy with limited interference between robots. This strategy performed equally well as *Divide and Conquer* in the second series of

experiments. We believe that it might outperform *Divide and Conquer* when the robot team size is significantly increased.

The experiments indicate that a *Divide and Conquer* type collaboration strategy is more effective at exploring an unknown environment than a *Reserves* strategy. Many existing multi-robot exploration and mapping schemes use a collaboration strategy which is similar to the *Reserves* strategy and might benefit from a *Divide and Conquer* based approach.

3.5 Discussion

Collaborative multi-robot systems are a useful approach to many applications of robots. Multi-robot systems can work together in the home to perform tasks such as meal preparation and moving heavy or cumbersome objects. Multi-robot systems are also useful for military counter-insurgency operations or for first-responders in disaster areas due to availability and redundancy.

We have presented how the *OmniMapper* system has been extended to work with multi-robot systems. The applications in this chapter have addressed how robots can work together to build a map via a simple recruiting strategy and a distributed data fusion (DDF) algorithm. A second study was presented which demonstrated that heterogenous teams of robots can compliment each other. In this case, a robot equipped with a sophisticated 3D laser scanner worked together with a simpler robot equipped with a 2D scanner to build a map of an environment. Finally, a study was presented which explored the strategy used for collaboration in an exploration and mapping task. This study presented some strategies by which a team of robots can increase performance on the task of exploring an unknown environment by collaborating together. This study focused on exploration and mapping; however, these strategies could be applied to any team of robots operating in an unknown environment where availability of team members must be traded off against interference in

confined spaces.

The next chapter will present how high level *semantic* information can be used by a robot to perform data association on landmarks in its environment. This is the key improvement associated with using objects as landmarks for mapping.

IV

OBJECT MAPPING

Up until now, we have described techniques that robots can use to build geometric maps of their environments. These maps are of value to robots for use in navigation as well as for sharing with humans for situational awareness or planning; however, a more sophisticated representation is needed to enable higher-level autonomous robot behavior. Semantic mapping deals with incorporating and understanding deeper meaning associated with the landmarks in the environment, such as:

- Walls divide rooms in a house
- Doors lead to other spaces when opened
- Objects are related to each other
- Rooms are related to each other
- Objects are related to rooms

Understanding these types of semantics addresses some of the challenges in mapping such as perceptual aliasing and the *kidnapped robot problem*. In both of these cases, the robot becomes lost and must *re-localize* itself in the environment. A robot can use a semantic map to recover from these conditions.

Objects are important cues for mobile robot operation in a man-made environment. Some objects serve as distinctive permanent landmarks for navigation, such as major appliances, paintings hung on the wall, and furniture; while other objects need to be tracked and their location memorized, such as the TV remote, cooking utensils, and a bottle of the owner's favorite beverage.

There are many different kinds of objects in most environments. At the same time general environments such as office buildings have a significant number of signs designed for human navigation such as exit signs, number signs for offices, fire hydrant indicators, and defibrillators to rescue heart attack victims. It is consequently of interest to explore how such signs can be utilized by robots to simplify their navigation tasks.

Object recognition, as mentioned earlier, is a widely studied subject in computer vision [Fergus et al., 2003]. The typical process is one of doing feature extraction, match features against a database to determine identify. For robots it is of interest to consider selection of computationally efficient feature descriptors. Two of the popular feature descriptors are SURF and SIFT. We analyzed these feature descriptors for their performance on object recognition and classification tasks in a study, which is presented in appendix 8.1. An alternative to a specific feature selection is to use machine learning to automatically design decision surfaces for discrimination between a variety of signs/objects. In section 4.1, a study comparing Support Vector Machines and Relevance Vector Machines for recognition is performed. In section 4.2, we present how these techniques can be embedded in a mapping framework to simplify data association and in particular improve loop closing.

More recently, a new generation of 3D sensors such as the Microsoft Kinect have been introduced. These sensors simplify the figure-ground segmentation problem, and also provide a rich set of features. The 3D point cloud from these sensors has a rather limited spatial resolution; to compensate for this we describe in section 4.3 how the point cloud data can be complemented with high-resolution *foveated* data from a regular camera.

Recognizing and classifying objects is an important open problem in computer vision. The first part of this chapter in section 8.1 is a study we performed to determine a viable technique to perform two related tasks, object recognition and object

classification. Two of the most important feature descriptors, SIFT and SURF are compared on a data set for both recognition and classification performance.

One of the most semantically relevant object types is a sign. Signs are designed to convey meaning about navigation in indoor and outdoor settings. The second part of this chapter, section 4.1, describes a machine learning technique to recognize door signs in an office environment. Signs convey semantic meaning about places and door signs provide navigational cues with regard to the room to which they are adjacent. Door signs are used for mapping and *semantic data association* in section 4.2. The door signs that are mapped contain an important cue: a string of numbers uniquely identifying the room adjacent to it. This cue will be used to perform *semantic data association*, which re-localizes a robot after it becomes lost. This type of reasoning will be extended to general objects in chapters 5 and 6.

Domestic service robots need to be able to find and recognize the objects which they use to perform their duties. Newly developed commodity 3D cameras are used with high resolution *foveated* cameras for recognition and classification in section 4.3. These components are used to identify objects in chapters 5 and 6.

4.1 Office sign recognition with Relevance Vector Machines[1]

Design of intelligent systems for operation in indoor environments is an important challenge for the future. The ability to read signs is an important competence to achieve such a vision. A system that can detect office signs is presented in this section. Histogram of Oriented Gradients (HOG) features are used to train Relevance Vector Machines (RVMs) and Support Vector Machines (SVMs). Performance on office signs from several buildings is compared using both classifiers. HOG combined with RVMs offers a compelling solution to the problem.

[1] ***This section is from an unpublished project report prepared with my colleagues Alex Trevor and Carlos Nieto***

4.1.1 Introduction

While roboticists often frown on the use of artificial landmarks for robot localization tasks, we might be overlooking the fact that humans already augment their environments with certain artificial landmarks to facilitate navigation tasks. Buildings generally contain a significant number of signs, such as those on doors that indicate a room number, and perhaps also a room name, or the name of the room's owner. Additionally, signs are often placed in hallways that indicate which direction to travel in order to reach various locations. Since these signs serve as excellent landmarks for humans, shouldn't robots also take advantage of them?

As a first step towards making robots capable of using signs, we have developed a classifier capable of detecting whether an image patch is a door sign or not. Our approach is to extract Histogram of Oriented Gradient (HOG) features from candidate sign regions and compare the classification results with Support Vector Machines (SVMs) and Relevance Vector Machines (RVMs).

4.1.2 Related Work

Previous work has been done on recognition of road signs for robot navigation. In de la Escalera et al. [2004], the authors first segment sign-like regions based on color and shape. These regions are then analyzed by a set of SVMs which are trained on each type of road sign. In this application, the specific type of sign must be determined, so many separate SVMs are trained for each type. More recently, Maldonaldo-Bascón et al. [2007] has implemented a similar system which also classifies the sign-like regions with an additional SVM using the Distance to Borders histogram descriptor.

RVMs are a reformulation of SVMs with Bayesian reasoning which delivers a probability distribution as an output and also greatly reduces the number of support vectors and tuning parameters [Tipping, 2001]. RVMs have been used for object tracking in Williams et al. [2003] by using their probabilistic output as a measurement.

RVMs are also used for 3D articulated body state estimation from monocular images in Thayananthan et al. [2005].

4.1.3 Approach

Our approach is to detect signs in images based on Histogram of Oriented Gradient (HOG) features. Many example images were annotated by hand to select image regions as either positive or negative examples of signs. For each of these examples, a HOG descriptor is calculated. These are then used to train both Support Vector Machines (SVMs) and Relevance Vector Machines (RVMs) to perform a binary classification problem: whether or not the image region was a sign.

We chose this approach because we believe that perhaps the most salient part of door signs are the edges. These signs are designed to be visually distinct from their surroundings, so we expect that they will be bounded by strong edges. Descriptors based on histograms of such edges seemed like a good fit for this type of detection. These features have been demonstrated to be successful at person detection in conjunction with SVMs in Dalal and Triggs [2005b]. In this section, we use a similar technique for the office sign detection problem.

4.1.3.1 Histograms of Oriented Gradient (HOG) Features

We selected HOG feature parameters to maximize recognition performance using coordinate ascent. HOG feature parameters which can be adjusted include the number of bins (edge filter orientations), the size of the detection window, and the contrast normalization window selection. We chose 16x16 pixel window size and 4x4 histogram cells per window and 9 orientations through 3-fold hold-out cross validation experiments. We did not detect an advantage for varying the overlapping contrast normalization regions in our application[2].

[2]The HOG feature implementation from OpenCV Bradski and Kaehler [2008] was used for this module.

Histograms of Oriented Gradients features [Dalal and Triggs, 2005b] for the application of human detection, are visually detected features that allow for appearance based recognition using edge gradients. This feature is constructed by first normalizing the image contrast in a series of overlapping regions covering the interest region. In our implementation, only one contrast normalization region is used. The image is passed through a bank of edge filters and the responses are binned into histograms based on the location in the window. The histograms correspond to a cell decomposition of the window. We empirically determined that 8x8 cells per window generated the best results.

HOG features are constructed from an image window by first normalizing image contrast over a pattern of overlapping sub-windows within the window. We have chosen to only use one contrast normalization window because we empirically determined that the door sign is small enough to not exhibit significant contrast variation. The image is convolved with a set of oriented edge filters. These edge filter responses are binned into histograms which are positioned in a regular grid covering the initial window.

HOG features have a number of parameters that can be adjusted, primarily relating to the number of bins and size of the detection window. We experimented with these by running 3-fold cross validation on some of our data with various parameter settings to determine what would perform well for our application. Although we performed our experiments using 16x16, 32x32, and 64x64 window sizes, we found the 64x64 window size to produce the best performance, so we provide results for this window size.

The HOG feature represents patterns of image gradients. Door signs are designed to be visually distinctive from the background to make them apparent to humans who are using them for navigation. Well designed signs typically exhibit sharp contrast from the background which generate strong edges under gradient analysis. HOG

features were used with SVMs for classification in Dalal and Triggs [2005a] where they were shown to perform well at recognizing people in images[3].

4.1.3.2 Support Vector Machines (SVMs)

SVMs are a discriminative classifier which finds the maximum margin decision boundary on a kernel function of the input feature. The result of this learning procedure is a sparse set of *support vectors* which are the only components from the training set needed to perform classification. These support vectors are the examples which are nearest to the decision boundary. Classification is performed by evaluating the kernel function between the HOG feature of a candidate image region and each of the support vectors according to equation 32. In this equation, $y(x)$ is the predicted label, n is the number of support vectors, ω_n is the weight of the n-th support vector, $k(\cdot, \cdot)$ is a kernel function and b is an offset.

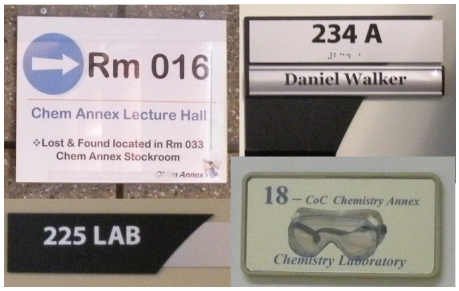
$$y(x) = \sum_{n=1}^{N} \omega_n k(x, x_n) + b \tag{32}$$

A variety of different kernel functions were tried for this recognition task including polynomials of degree 5 to 12, linear kernels, and the Radial basis function kernel (RBF). Polynomial kernels performed well in our cross-validation tests; however, the SVMs using them exhibited inferior generalization performance compared to the SVMs using RBF kernels. The γ parameter of the RBF kernel was selected via coordinate ascent to be 1.0.

The SVMs were trained with manually labeled regions extracted from images with our saliency technique detailed above; positive examples were provided which contained the door signs and negative examples were given of various other structures such as doorknobs, fire extinguishers, posters, door jambs, and random image regions.

[3]We used the HOG feature implementation included in the OpenCV Computer Vision and Machine Learning library [Bradski and Kaehler, 2008].

(a) Examples of signs used for training



(b) Examples of signs used for testing

Figure 40: Training and testing example images

4.1.3.3 Relevance Vector Machines (RVMs)

The RVM was introduced in Tipping [1999, 2001] as a reformulation of SVMs with Bayesian reasoning which delivers a probability distribution as an output while greatly reducing the number of support vectors and tuning parameters needed.

As with SVMs, we performed cross validation tests for RVMs to determine the optimal parameters for use in our application. The radial basis function has just one parameter, γ, which needs to be selected. We performed a coordinate ascent search over a small range of gamma values to determine what γ value performed well for our data sets.

We also briefly experimented with polynomial kernels for RVMs, and the preliminary results looked comparable to the RBF kernels. Polynomial kernels have three parameters to tune (gamma, degree, coefficient) instead of just one for RBFs, meaning we would need to do significantly more testing to determine a good set of parameters. Since RVMs take so long to train, we decided to only use the RBF kernels for the rest of our experiments[4].

4.1.4 Data Sets

We collected our own data sets of sign images to use for testing and training our system. The images were taken using a commercial digital SLR camera in several buildings on the Georgia Institute of Technology Atlanta campus. They were taken over the course of several days at various times of day, though most do not include windows that produce natural light. The images were generally taken from 1-2 meters away from the sign, facing the sign more or less directly. For many signs, additional images were taken from angles of 5-10 degrees in either direction. We chose these conditions because in our desired application these images will be taken from a robot which uses a laser scanner to position itself at an appropriate distance and angle relative to a doorway, allowing similar images to be taken by the robot's camera.

For our experiments, we partitioned these images into four data sets, representing different buildings. Note that each building can include a wide variety of sign appearances. We named these data sets based on the building names, which include "CoC", "Klaus", "Chem", and "TSRB". The data sets consists of 105 images, 133 images, 30 images, and 23 images respectively. From each image, approximately 10 to 20 positive and negative example regions were selected, and these were used as the input to the classifiers. Although each of these data sets represents a single building, the buildings contain multiple types of office signs, so each set contains multiple different appearances of signs. The CoC and Klaus data sets were used for training, while the Chem and TSRB data sets were reserved for testing the generalization of our detectors to other types of office signs.

The CoC data set consists of 105 images from the second floor of the College of Computing. This includes the RIM area offices, RIM cubicles, and the back part of the building, which includes an older style of signs. This also includes some signs on

[4]We used the D-Lib Machine Learning library's implementation of RVMs. More information on the D-Lib library is available in King [2009].

glass walls.

The Klaus data set includes 133 images from the Klaus building. This building includes several different types of signs, in varying amounts of clutter, so this is a fairly challenging data set.

The Chem data set consists of 30 images taken from the chemistry annex in the basement of the College of Computing, which contains a different style of signs from the rest of the building. This data set was used only for testing; no SVMs or RVMs were trained on these images. This allows us to test how well our classifiers generalize to other unseen buildings.

The TSRB data set consists of 23 images from the Technology Square Research Building (TSRB). It includes several types of signs, including signs from the GVU area, as well as the other offices. This data set was used only for testing; no SVMs or RVMs were trained on these images. This allows us to test how well our classifiers generalize to other unseen buildings.

4.1.5 Results

We performed two types of experiments for both SVMs and RVMs: K-fold cross validation, and a confusion matrix test to evaluate generalization performance.

We designed an application to allow us to manually provide bounding rectangles for both positive and negative examples of signs because our classifiers operate on image regions. For each data set, we generated a file containing many labeled regions per image.

SVMs and RVMs were trained on a total of three files: Lab-CoC, Lab-Klaus, Lab-CoC-Klaus. Lab-CoC contains examples drawn from the CoC data set, Lab-Klaus contains examples drawn from the Klaus data set, and Lab-CoC-Klaus is the concatenation of Lab-CoC and Lab-Klaus. The order of examples is randomized prior to training.

RVMs take a long time to train, so for some scales, the data was sub-sampled. Cross-validation and confusion matrix results for HOG scales 4 (16x16) and 5 (32x32) are given for 1500 bounding rectangle examples per data set. Scale 6 (64x64) was performed on the full number of samples for each data set, though these RVMs took several days to train.

For the confusion matrices, separate test files were used. Again, a test file was generated for each data set, but was labeled by a different person than the training set. In addition to the CoC and Klaus data sets, we also generated test label files for the additional building data sets of Chem and TSRB. These contained office signs with appearances quite different from the training sets, allowing us to test the classifier's ability to generalize to new types of office signs. Ds2, ds3, and ds4 are image regions drawn from data sets 2, 3, and 4 respectively, but these are different regions (labeled by a different person) than the training examples. So, although the source images are the same, the exact bounded regions are not the same as were used in training. DsChem and dstsrb were not used in any training sets, so they are new signs.

4.1.5.1 Cross Validation

The first experiment we performed was to do k-fold cross validation on each of our datasets. For our tests, we chose $k = 3$, so the classifier was trained on 2/3 of the data, and tested on the remaining 1/3 for three such partitions of the data. This allows us to determine how well the sign recognition works on other examples drawn from the same data set. It gives us an idea of how well this might perform on other signs of the same type (from the same building, for example). The SVM cross validation results are given in table 4.

As with SVMs, the same 3-fold cross validation was performed. The RVM cross validation results are given in table 5.

Dataset	Positive	Negative	Num Vectors
a:CoC-lite	0.625	1.000	82
b:CoC	0.975	0.998	218
c:Klaus	0.907	0.968	956
a & b	0.954	0.968	303
a & c	0.898	0.971	1003
b & c	0.918	0.984	1193
a & b & c	0.917	0.984	1206

Table 4: SVM cross validation results.

Dataset	Positive	Negative	Num Vectors
a:CoC-lite	0.708	0.947	9
b:CoC	0.970	0.993	15
c:Klaus	0.897	0.964	154
a & b	0.949	0.985	35
a & c	0.889	0.965	120
b & c	0.905	0.965	193
a & b & c	0.915	0.964	157

Table 5: RVM cross validation results.

4.1.5.2 Building Confusion Matrices

The SVMs we trained on each dataset (and each combination of data sets) were then tested on each of the test sets, as described above. The first number in each cell is the percentage of true signs classified as true, and the second number is the percentage of false signs classified as false. The results are given in table 6.

As with SVMs, a confusion matrix was calculated for RVMs as well. RVMs provide probabilistic classification of signs, so we needed to choose a threshold for signs vs. not signs. For these experiments, we treated any descriptors less than or equal to 0.5

	CoC		Klaus		Chem		TSRB	
CoC	0.98	1.00	0.02	0.99	0.00	1.00	0.18	1.00
Klaus	0.07	1.00	1.00	0.99	0.09	0.98	0.53	0.97
CoC & Klaus	0.98	1.00	1.00	0.99	0.09	0.98	0.58	1.00

Table 6: SVM confusion matrix results. CoC and Klaus represent combinations of training on the same building (but with different specific signs). Chem and TSRB are completely different buildings.

	CoC		Klaus		Chem		TSRB	
CoC	0.98	1.00	0.30	0.99	0.05	1.00	0.35	0.97
Klaus	0.14	0.96	0.99	0.98	0.09	0.98	0.48	0.97
CoC & Klaus	0.95	0.99	1.00	0.98	0.18	0.98	0.50	1.00

Table 7: RVM confusion matrix results. CoC and Klaus represent combinations of training on the same building (but with different specific signs). Chem and TSRB are completely different buildings.

as not signs, and anything greater than 0.5 as a sign. Adjusting this threshold could potentially improve results. The results are given in table 7.

4.1.6 Discussion & Conclusions

We can draw several conclusions based on our experiments, both about the use of HOG feature for recognition of signs, and about the use of SVMs and RVMs for classification based on HOG features.

We can see that as we train on more and more different types of signs, we get better generalization, even to types of signs that we didn't train on at all. This can be seen clearly in the confusion matrices both for SVMs in table 6 and RVMs in table 7. The *CoC-Klaus* classifier outperforms both the *CoC* and *Klaus* classifier on the unseen data sets in these tables.

Overall, both SVMs and RVMs seemed to be appropriate choices for this task. SVMs train very quickly, but keep a significant portion of the data as support vectors. On the other hand, RVMs take an extremely long time to train, but produce comparable results to SVMs using many fewer support vectors. RVMs have the advantage of providing probabilistic classification, which can be used with tracking to improve performance.

In conclusion, we found that HOG features are a good fit for recognition of signs, and seem to capture a good amount of the discriminating information for this task. We also found that both SVMs and RVMs can perform quite well for this task, and generally have fairly comparable performance. Our preliminary system had promising

results, and suggested some potential future work as described above.

4.2 SLAM with Learned Object Recognition and Semantic Data Association[5]

Complex and structured landmarks like objects have many advantages over low-level image features for semantic mapping. Low level features such as image corners suffer from occlusion boundaries, ambiguous data association, imaging artifacts, and viewpoint dependance. Artificial landmarks are an unsatisfactory alternative because they must be placed in the environment solely for the robot's benefit. Human environments contain many objects which can serve as suitable landmarks for robot navigation such as signs, objects, and furniture. Maps based on high level features which are identified by a learned classifier could better inform tasks such as semantic mapping and mobile manipulation. In this section, we present a technique for recognizing door signs using a learned classifier as one example of this approach, and demonstrate their use in a graphical SLAM framework with data association provided by reasoning about the semantic meaning of the sign.

4.2.1 Introduction

Bridging the gap between mobile robots operating in the factory and operating in everyday environments requires the development of SLAM techniques and semantic reasoning. The inclusion of object-level landmarks in maps facilitate tasks such as object retrieval and more generalized human robot interaction dialog.

Complex and structured landmarks such as objects have many advantages over low-level image features for semantic mapping. Low-level features suffer from viewpoint dependent imaging conditions such as boundary occlusion (where the feature is on a boundary and will appear different in subsequent frames due to motion parallax), insufficient invariance to robot motion, and specular reflections.

[5] ***This section is adapted from our paper in IROS 2011 Rogers III et al. [2011].***

Data association is a problem for most SLAM algorithms operating in unstructured environments. Low-level features make use of validation gates and joint compatibility to mitigate this problem; however, the use of higher level features reduces the significance of this problem, since each landmark might have uniquely identifiable characteristics. Signs, for example, often contain text which can be read by the robot to give a unique string which could be used as an unambiguous data association cue.

Semantic mapping also offers an advantage for robots to understand task assignments given to them by human users. Non-technical users will prefer human terms for objects and locations when assigning tasks to robots instead of whatever indices or coordinates the robot uses to represent them in its memory. A text string which could be read from a sign, such as "Room 213" provides semantic information both as a label, associating "213" with the present region and denoting the place as a "room".

In this section, we present a method for using a learned object classifier in a SLAM context to provide measurements suitable for mapping. To demonstrate this, we present a classifier for recognizing door signs and a data association technique based on reading text in a graphical SLAM framework for an office environment.

Related work will be presented in section 4.2.2. The specific algorithms and techniques used in this paper will be presented in section 4.2.3 and section 4.2.4. The experimental procedure will be outlined in section 4.2.5. Results will be shown in section 4.2.6 along with some additional techniques which were developed to improve results. Conclusions will be presented along with our future plans in section 4.2.7.

4.2.2 Related Work

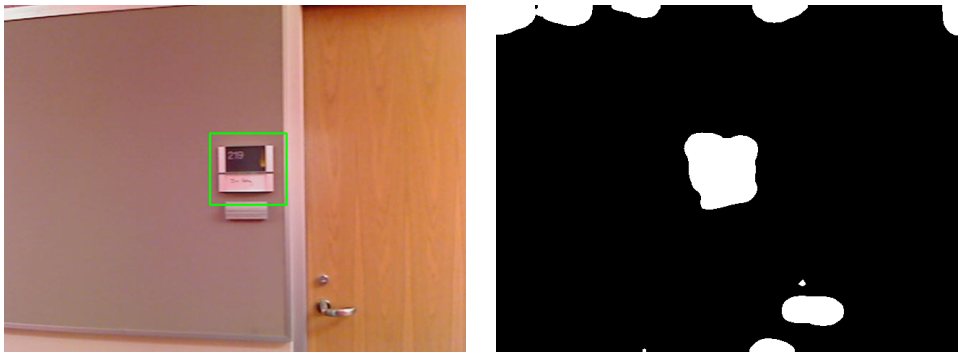
Castle et al. [2007] incorporated known planar objects as part of visual SLAM. This technique extracts SIFT features [Lowe, 2004b] from the image and periodically finds inliers to a homography from a canonical view of the known objects. Our work differs from this technique in that the object recognition module is not finding matches to

a small set of known objects but is based on classification of objects which may not have been seen before.

Recognition and reading of door signs was demonstrated in Tomono and Yuta [2000]. This work recognized and read specific door signs in a building and estimated their relative pose with respect to the robot. More recently Tomono *et. al.* have developed an object recognition scheme which they used with a mapper to build object maps [Tomono and Yuta, 2003]. In contrast to this work, our approach uses a machine learning technique for recognition which could be extended to other types of objects.

4.2.3 Door Sign Detection

To demonstrate our technique for mapping using learned object classifiers, we selected door signs as these are landmarks that frequently appear in human environments. They provide strong cues for navigation to humans, and can do so for robots as well. We used the method for visual door sign detection described in section 4.1.3.



(a) Example image seen by robot (b) Saliency mask

Figure 41: An image of a door sign seen by the robot is shown in figure 41(a) and the resulting saliency mask is shown in figure 41(b)

To enable efficient online operation, a method for selecting image regions as candidate door signs for our classifier was developed. The first step is to identify regions of the image which are visually distinctive. The spectral residual saliency technique

described in Hou and Zhang [2007] is used due to its straightforward implementation and acceptable performance. A typical saliency result image is shown in figure 41(b). The saliency image is analyzed to find blobs which are of appropriate size to be candidate sign regions. Occasionally, the saliency technique fails to isolate the sign, when the scene is too complex. For this reason, we have also added a series of fixed regions which cover the image frame at different sizes. This provides more chances to find the door signs when the saliency technique fails.

The door sign detection module uses the Histogram of Oriented Gradients feature for recognition [Dalal and Triggs, 2005a], as described in section 4.1.3. Candidate door sign regions extracted using our saliency technique were hand-labeled as signs or not signs, and rescaled to a square 16x16 pixel size. A HOG feature was extracted from this region. A Support Vector Machine (SVM) was then trained on these features.

4.2.3.1 Optical Character Recognition

Door signs in our building contain a unique recognition cue – a number and/or name identifying the room beyond the adjacent door. We use this cue for data association by reading the sign using a request to the GoogleGoggles server. We had previously attempted to use the open source optical character recognition (OCR) software library called *Tesseract* [Smith, 2007], but we found that while it works very well on analyzing scanned printed black-on-white text, it is difficult to adapt to camera images of text on signs.

Despite the significant performance improvements achieved by leveraging the GoogleGoggles service, sometimes the text strings returned contain a few errors which must be handled before they can be used for data association. These errors typically arise from the service matching non-text graphics or borders to a similarly shaped character, as well as missing text due to over or under segmentation due to lighting variability. To cope with these minor mistakes in reading, we first split the number

component and the text string. A match in the number component results in a positive match. If the number fails to match, then the text string component is stripped of all whitespace and converted to lower case. The longest common subsequence is extracted between the mapped sign and the current measurement. If this subsequence is more than about 60% of the length of the longest of the two strings, then the text string results in a positive match.

4.2.3.2 Training

We trained classifiers on datasets from two different buildings on the Georgia Institute of Technology campus: the College of Computing building and the Klaus building. These datasets contained 105 and 133 images respectively, and approximately 10 to 20 positive and negative example regions were manually specified in each image as the training examples. Several example images from our training set are shown in figure 42. 3-fold cross validation was performed for each classifier, and the results are summarized in table 8. The resulting SVMs were also tested on separate test data sets both from the same buildings, shown in table 9. While the classifiers have a low true positive rate for styles of signs that they weren't trained on, the false positive rate also remains low. The performance is a good match for use in our SLAM application, because missing a few true positives is acceptable, but adding false positives is highly undesirable. Also, we believe that the classifier's performance on the style of signs it has been trained on is more important than generalization to unseen sign types, because it is not unreasonable to imagine a robot that requires some amount of training specific to its intended environment. For the mapping experiments in this paper, we trained the classifier on images of the same types of signs as the test set from another part of the building. To be used for mapping, a sign which is selected by this classifier must also contain text which is understood and returned by GoogleGoggles. This condition further reduces the false positive rate so that no false positives have

Dataset	Positive	Negative	Num Vectors
CoC	0.953	0.985	262
Klaus	0.774	.963	1195
CoC-Klaus	0.795	0.955	1495

Table 8: Door Sign classifier SVM cross validation results.

	CoC		Klaus	
CoC	0.954	0.960	0.321	0.984
Klaus	0.22	0.939	0.915	0.969
CoC-Klaus	0.908	0.949	0.915	0.974

Table 9: SVM confusion matrix results on trained buildings. True positive rate is listed on the left, true negative rate is listed on the right.

been observed during our testing.

4.2.4 Mapping

Our robot makes use of the Robot Operating System (ROS) developed by Willow Garage [Quigley et al., 2009] for control of the flow of data. Our technique uses three new software modules: the *laser-line-extractor*, the *door-sign-detector*, and the *mapper*. The *laser-line-extractor* was explained in section 2.4.2, and the *door-sign-detector* will be described in the following section.

4.2.4.1 Door-sign-detector

The door-sign-detector module makes use of the classifier described in Section 4.2.3 to recognize door signs in images taken from the robot's camera. If an image region is classified as a sign by the SVM, then a query is made from this image region to the GoogleGoggles server. If GoogleGoggles is able to read any text on the sign, then it will be returned to us in a response packet. Detected signs with decoded text are then published as measurements that can be used by the mapper. The measurements consist of the pixel location in the image of the detected region's centroid, the image patch corresponding to the detected region, and the text string returned from GoogleGoggles.

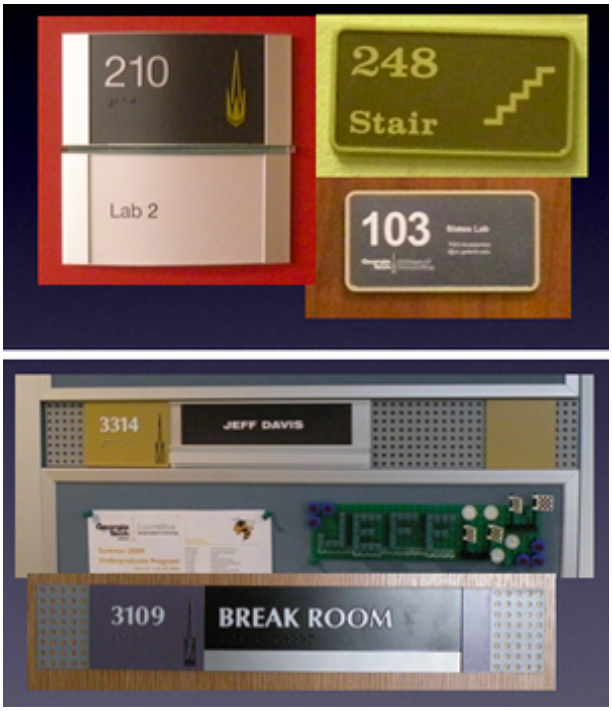


Figure 42: Examples images of signs from our classifier's training set. Signs on the top are from the College of Computing dataset, and signs on the bottom are from the Klaus data set.

4.2.4.2 Mapper

The mapper used in this study is the *OmniMapper*, which has been introduced in section 2.4. Briefly, it is a complete SLAM solution based upon the factor-graph based nonlinear optimization engine *GTsam*. *OmniMapper* extends *GTsam* with the Measurement space (M-space) feature representation developed by Folkesson *et. al.* [Folkesson et al., 2005b,a, 2007]. This representation allows us to use different types of features such as walls and signs in a unified framework.

Originally, measurements of door sign features were implemented as projections of mapped features into image coordinates and direct comparison of pixel error from measured values. This approach proved unstable to initialize and did not work well when it was used to close extremely large loops with significant error. Measurements are now made on the 3D coordinates of the back-projected image location directly. Range is recovered by finding the laser beam from the head laser which is

projects most closely to the image coordinates of the sign. This technique approximates the true range which we will eventually get from the use of a 3D camera like the Kinect. This factor also incorporates an additional variable which corresponds to the transformation between the robot base and the camera. By keeping track of the transformation when each measurement is taken, we are now able to move the camera on the pan-tilt unit during a data collection run.

To implement this factor in GTSAM, we must specify an error function and the error function's derivatives in terms of all of the variables which contribute to it. The error function is the difference in the 3D position of the predicted location of the sign from the measured value given by the recognition module. In equation 33, the error function is computed in terms of the robot pose x_r, the transformation from the base to the camera T_{bc}, and the pose of the landmark x_f. The measured location of the sign in the robot reference frame is z.

$$h(x_r, T_{bc}, x_f) = z - T_{bc}^{-1} * x_r.transform\_to(x_f) \tag{33}$$

The three Jacobians of this error function are computed by combining various primitive derivative operations using the chain rule. The Jacobian of the error function with respect to the robot pose, $\frac{\delta h}{\delta x_r}$ is found as the product of the derivative of the projection operation from the camera with respect to the camera pose, times the derivative of the 3D pose composition operator with respect to its first entry. The second Jacobian, $\frac{\delta h}{\delta T_{bc}}$ is the same as the first Jacobian, with the pose composition derivative taken with respect to the second parameter. The final Jacobian, $\frac{\delta h}{\delta x_f}$ is just the derivative of the projection operation with respect to the object pose.

4.2.5 Experiment

We performed a series of experiments to demonstrate the use of a learned classifier to generate landmark measurements in a SLAM context. We used the door sign

classifier described in section 4.2.3 to generate measurements from door signs in our office. The classifier used in this experiment was trained on images taken from a hand-held camera from a variety of different door signs on the second floor of our building. Multiple test runs consisting of different size loops were collected. Additionally, the training set was made from hand labeled and selected regions while the test run was made using the automatic saliency analysis and blob extraction and fixed sampling as explained in section 4.2.3. We also collected wall measurements from the laser scanner and used both feature types to generate maps.

Figure 43: The Segway RMP 200 with LMS 291 laser scanner for wall measurements and a Prosilica 650c camera with a Hokuyo UTM30 laser scanner on a PTU-46-70 pan-tilt unit. Four caster wheels were added for stability. The robot is shown in a position typical of reading a door sign.

The robot is shown in figure 43. It is a Segway RMP-200 modified with external caster wheels. This modification allows us to operate without using the balancing mode, which offers additional stability and safety. The robot makes use of a SICK LMS-291 laser scanner to collect measurements of walls and a Prosilica 650c camera with a Hokuyo UTM30 laser scanner mounted on a pan-tilt unit to collect images of door signs. The pan-tilt unit was controlled by the robot operator to point at the

door signs during the test data collection. Camera images are collected automatically at a regular interval, not just when the camera is aimed at a door sign.

Currently, the robot is tele-operated in the environment while its sensor and odometry data are logged by ROS. The data file is processed offline by our system to construct the final map. This is done for convenience and repeatability, as well as to support our development. Our algorithms run in better than 2x real time with up to 353 poses and over 800 total measurements in the longest run. The transaction through GoogleGoggles server however does require about 4 seconds per frame, but this is only performed on image regions which are determined to be door signs. The mapper has been designed to operate asynchronously with measurement sources, so it can process messages arriving out of order from the recent past.

4.2.6 Results

Figure 44: This sign is recognized and a measurement is made in the mapper. GoogleGoggles has read both the room number and the text, so this sign can be used for data association.

A total of five test runs were performed, two runs at each of short and middle loop sizes, and one run at the large loop size. An example image is shown in figure 45, which illustrates the types of features which are mapped and how they are displayed in the maps.

The short loop size is about 30 meters. In the runs at this loop size, the robot starts by proceeding down the west hallway and it is carefully driven and the camera

is aimed at the door signs. The robot then drives through a cluttered laboratory space where few measurements can be made of the walls, resulting in significant pose error when the robot exits the lab. The robot is then driven back into the west hallway, but it doubles the hallway because of significant pose error, see figure 46(a). In each of the test runs, the robot makes three successful sign matches and realigns the map, as shown in figure 46(b).

The middle size loop run takes the robot first down the west hallway, and then onward into the half of the floor which is still under construction. Significant portions of this area contain clutter and are difficult to find wall segments large enough for mapping, resulting in some significant pose error in this portion. The robot then proceeds through the back hallways and around the loop for about 200 meters, where it re-enters the west hallway. The robot is now lost because the walls are not successfully matched due to significant pose error, see figure 47(a). After door signs are matched, the robot is relocalized and the map is corrected in figure 47(b).

The long run starts out the same as the middle length run but instead of proceeding back to the west hallway after exiting the back hallways and the construction area, the robot proceeds around the largest loop possible on our floor by driving down the east hallways and through the kitchen and atrium before returning to the west hallway. As with the other runs, the robot has become lost by the time it re-enters the west hallway and is unable to close the loop using only the wall features, see figure 48(a). Once again, door signs are re-observed and the loop is closed, resulting in a useable map and a localized robot, see figure 48(b).

4.2.7 Conclusion

We have shown that a learned feature classifier can be used to detect objects which can then be mapped in SLAM. Specifically, we have demonstrated the use of an SVM to classify HOG features for mapping in graphical SLAM. The door signs selected as

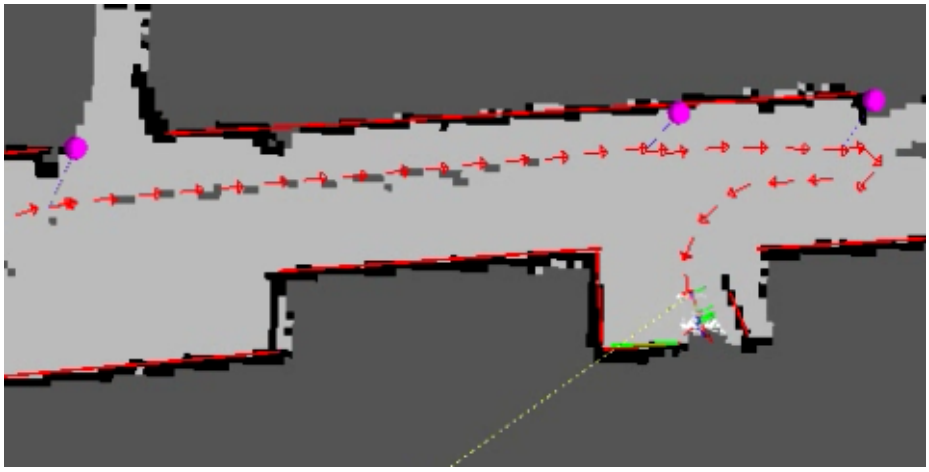
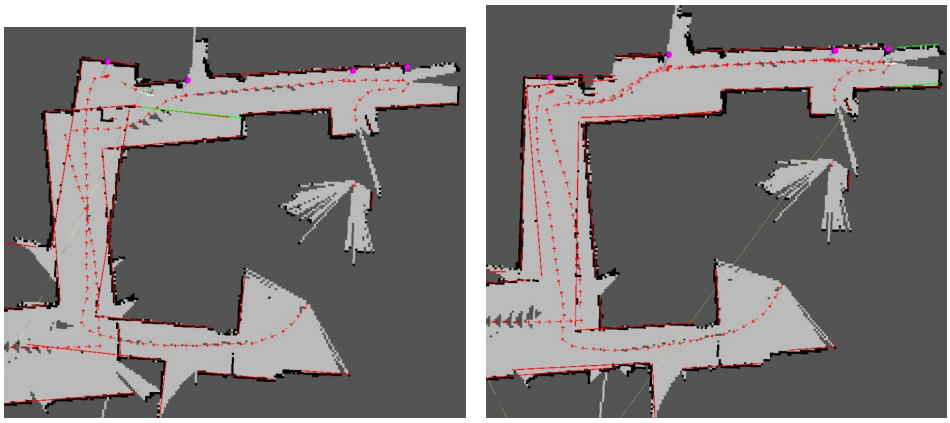


Figure 45: A close-up of the west hallway. Robot poses are shown as red arrows. Wall features are shown as red lines. Door signs are shown as pink spheres. The occupancy grid is displayed only for clarity. This figure is best viewed in color.

the object for detection possess semantic information, the text describing the room beyond the door, which we use for data association to close large loops, relocalizing the robot when it was lost. The understanding of the semantic meaning of these sophisticated features enabled the robot to map in a large and complex environment where it would have otherwise become lost.

The HOG feature classifier could potentially be used to recognize many different types of objects in the environment such as appliances, posters, and perhaps furniture. In addition to these *appearance based* techniques for object recognition, we could also consider using *model based* techniques which take into account the 3D model of the object being recognized. We wanted to leverage the generalization performance of appearance based techniques to recognize objects which have not been seen before. If we instead were able to establish correspondences between the image and a 3D object's pose through the use of a known CAD model, then a *pose measurement* could be given to the mapper instead of a less constraining *point measurement*. This technique could be used to recognize a small set of known objects but might not offer as much in terms of generalization to unseen examples of a class of objects, like door signs.



(a) Robot becomes lost in cluttered lab.

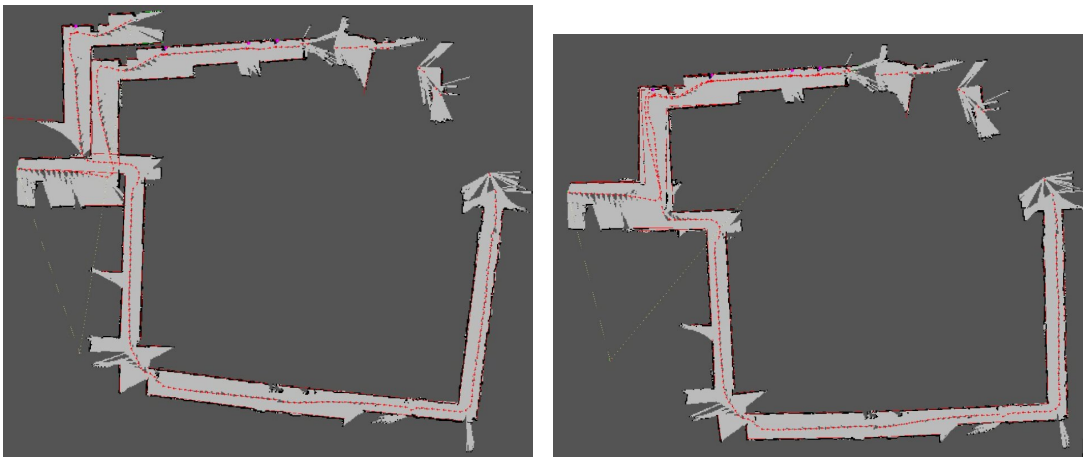
(b) A door sign is re-observed and the text is matched, resulting in a loop closure.

Figure 46: The robot is driven through a cluttered lab where it becomes lost 46(a). The robot recognizes and semantically data associates a previously seen sign and fixes the map 46(b)

Signs are specific examples of objects which convey a great deal of information beyond just their position for localization. Most objects which would be of sufficient permanence and importance to be considered as valid for mapping would also probably be relevant to other tasks that the robot might be required to perform. An example would be finding appliances would inform the robot that it is in the kitchen, so it would know where to find other food preparation equipment. The door sign features with room numbers could also be used to identify locations in a semantic map for understanding human commands.

4.3 Object recognition and classification

This section will describe the new components developed for performing object recognition and classification. These components are used in chapters 5 and 6 to provide information about the objects in rooms.



(a) The robot travels a large distance, resulting in a several meter trajectory error.

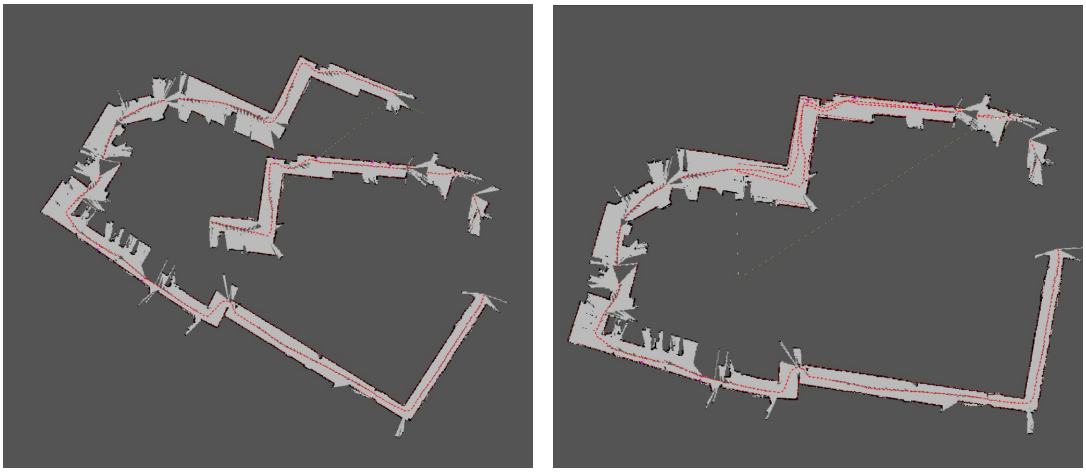
(b) Once a door sign is re-observed, the loop is closed

Figure 47: A longer mapping run through the hallways in our building. The mapping run goes through a cluttered area under construction and gets lost 47(a). A door sign is recognized, the loop is closed, and the map is corrected 47(b)

4.3.1 Object Segmentation

The first step in recognizing or classifying an object is to segment it from the background. Prior to the development of commodity 3D cameras, figure-ground segmentation was an active research subject.

Objects are segmented from the background with 3D point clouds. We leverage the Asus Xtion Pro depth camera (which has a Primesense sensor similar to the Microsoft Kinect) to observe the 3D structure of the scene immediately in front of the robot. The camera software provides a 3D point cloud which we operate on with the Point Cloud Library (PCL)[Rusu and Cousins, 2011]. First, the point cloud is spatially filtered to contain only relevant portions which fall within a volume of interest in front of the robot. The point cloud is then downsampled to one point per cubic centimeter using a voxel filter. We then extract up to 4 planes from the remaining point cloud using a RANSAC [Fischler and Bolles, 1981] technique available from the PCL library. These planes are then analyzed to find remaining points which lie above them. These remaining points are clustered to find candidate objects. The points



(a) In the long loop test, there is significant trajectory error before the door signs are re-observed.

(b) The robot re-observes a door sign and the map is corrected. The robot now knows where it is.

Figure 48: The robot has completed the long loop around the building, but has not yet found a sign match to perform a loop closure 48(a). Further along 48(b), the robot re-observes a door sign and the map is corrected.

from each candidate object of appropriate size are projected into a high resolution camera image which was taken at the same time. The extent of these points in the camera image is used to segment the candidate object from the background. This segmented image region is then passed on to the subsequent components to perform object recognition or classification. An example selected region of interest, object points, and table surface points can be seen projected into a high resolution image in figure 49.

4.3.2 Object Recognition

After an candidate object image is segmented using the technique described in section 4.3.1, we attempt to recognize it using a SURF feature [Bay et al., 2006] matching technique. First, the SURF features in the candidate object image are extracted. These SURF features are compared to the SURF features previously extracted in a model database. The set of matched features to a model element are used to compute a homography to the model image using RANSAC [Fischler and Bolles, 1981]. This

Figure 49: Point cloud data is projected into the camera image. Horizontal planes are extracted (yellow points) and objects are clustered points which appear above the plane (green points). The region of interest is selected based upon the projection of object points into the image (blue rectangle)

	Cooking	Electronics	Food	Sundries	Tools	Toys	None
Cooking	32	0	0	0	0	0	0
Electronics	0	23	3	0	0	6	0
Food	0	0	32	0	0	0	0
Sundries	0	0	1	31	0	0	0
Tools	0	4	1	0	24	3	0
Toys	0	0	0	0	0	32	0
Background	0	0	0	0	0	14	0

Table 10: Recognition test experiment without validation. Precision: 84%. Recall: 84%

homography is then used to filter out erroneous feature matches and select an inlier set. If this set is of sufficient size, then the object is said to be recognized absolutely and this information is then sent to the mapping system along with geometric measurements of the object's location. If there are not enough inlier feature matches, then the object is not recognized and must be classified by the next module described in section 4.3.3. The use of a homography is only fully correct when the candidate object is planar; further developments include 3D feature coordinates and matches consistent with camera projections. An example of a correctly identified object can be seen in figure 50(a). An example where this technique identifies a non-matching object can be seen in figure 50(b).

(a) A successfully recognized object from the class "Toys".

(b) An object which does not match with class "Sundries".

Figure 50: The model image appears on the right, the candidate object image appears on the left. Blue lines indicate matches which are inliers to the homography filter. At least 14 features must be matched for the recognition system to identify an object.

	Cooking	Electronics	Food	Sundries	Tools	Toys	None
Cooking	32	0	0	0	0	0	0
Electronics	1	17	0	0	1	0	13
Food	0	0	32	0	0	0	0
Sundries	0	0	0	31	0	0	1
Tools	0	0	0	0	22	0	10
Toys	0	1	0	0	0	28	3
Background	0	0	0	0	0	0	14

Table 11: Recognition test experiment with homography validation. Precision: 98%. Recall: 85% (including background class matching to "None")

(a) Cooking (b) Electronics

(c) Food (d) Sundries

(e) Tools (f) Toys

Figure 51: The set of object instances used in the recognition test. These object instance images are taken approximately 45° apart. For each instance view, a test recognition is performed by matching it against all remaining images.

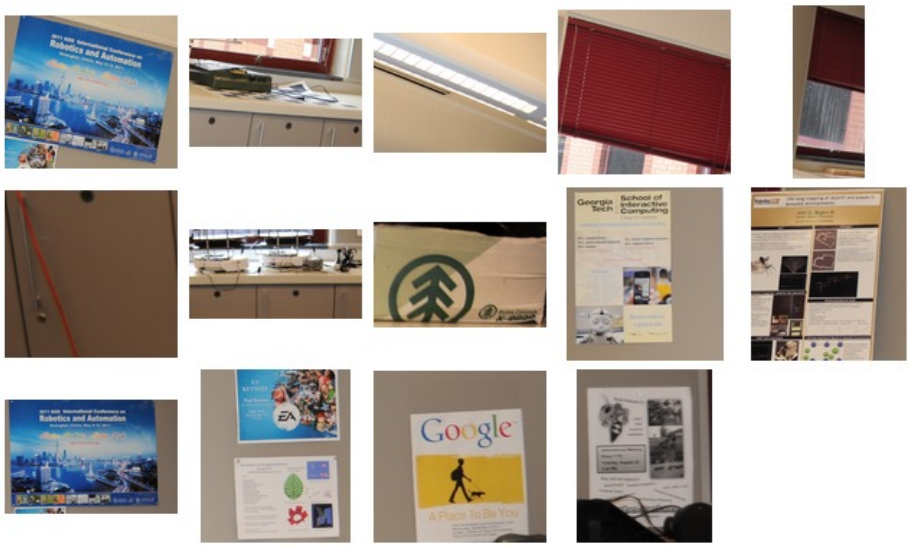


Figure 52: Images from the background that do not contain any modeled object instance.

An object recognition test was performed to establish the effectiveness of the recognition system. For this recognition test, an object database was created with four sample objects in each of six object classes (Cooking, Electronics, Food, Sundries, Tools, and Toys). The object database contains 8 canonical views of each object, separated by 45° in view angle. The test object database can be seen in figure 51.

To evaluate the performance of the recognition system, each of the images in the database are matched against all other images via the SURF feature comparison detailed in this section. In addition to the images in the object database, an additional set of background images, shown in figure 52, are tested against the object database. These images are not present in the database, so a match to one of the objects from these background images would constitute an error in recognition.

The first evaluation of the recognition system was performed with the homography-based geometric validation step turned off. In this test, the object in the database with the largest number of feature matches to the test object that pass a bi-directional ratio test is said to be the matched object. The bi-directional ratio test requires that each feature match be the best in both directions, from the model image in the

database to the test image, and vice-versa. This feature match also pass a ratio test; it must be more than 20% better than any other potential feature match. If this procedure results in at least 15 features matched, then the match is accepted. If this fails to happen, the recognition procedure rejects this match and returns that the object is of the "None" class.

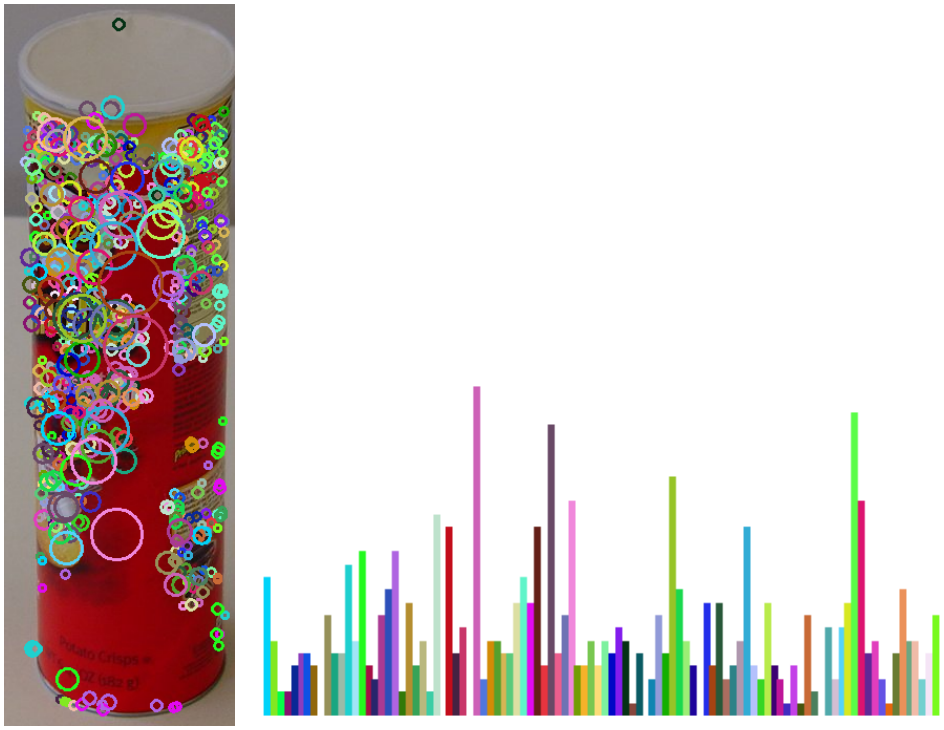
The result of the first test without the validation step can be seen in table 10. In every operation, the test image is matched to an element of the database; there are no "None" results, even amongst the "Background" test images from figure 52. Most of the mistakes which are made in this recognition test consist of an erroneous match to the Toys class. This error is likely due to the presence of a complex object instance of the "Macross" figure, which can be seen in the third and fourth rows of figure 51(f). This object is highly detailed and contains many features; this feature diversity allows for matches to simpler objects from the database when a geometric validation step is not used. The recognition results from this experiment are an 84% precision and recall.

The second evaluation is the same as the first one detailed above, with the addition of a homography-based geometric validation step. In addition to passing the bi-directional ratio test, feature matches must align in a geometrically consistent relationship with an image from the database. This geometric validation is performed through a RANSAC [Fischler and Bolles, 1981] homography test. The results of this experiment can be seen in table 11. The "Background" images are all correctly recognized as not containing an element of the database. Most of the mistaken recognitions are now moved into the "None" class, and are failures to recognize instead of hallucinated members of the wrong class. Overall, this validation step increases the precision to 98%, while maintaining a recall of 85%. The quoted precision includes the successful classification of "Background" images as being in the "None" class.

4.3.3 Object Classification

If a candidate object cannot be recognized by any of the models using the object recognition technique described in section 4.3.2, then we employ a probabilistic object classification technique to give a distribution over classes. The technique for object classification is based upon the *bag of visual words* approach of Sivic and Zisserman [2005]. This technique requires a vocabulary of visual words which was learned by performing K-means clustering on SURF descriptors extracted from the *Caltech 101* [Fei-Fei et al., 2004] training set. SURF features are extracted from an image of a candidate object and are vector-quantized to the nearest visual word in the vocabulary. The visual words extracted from a candidate object image can be seen in figure 53(a). A histogram is built by counting the frequency of the appearance of each visual word in this image of the candidate object weighted by the term frequency inverse document frequency (TF-IDF) [Ramos, 2003], see figure 53(b) for the histogram generated from the visual words seen in figure 53(a). A set of Relevance Vector Machines (RVMs) are trained to recognize object categories using this visual word histogram. Each of the RVMs is trained to recognize one category. The output of an RVM is a probability, unlike the output of a Support Vector Machine (SVM).

To perform object categorization on a segmented candidate object image, SURF features are first extracted and then vector quantized into their nearest visual word. The visual words present in this image are collected in a histogram and normalized. The set of RVMs representing the object categories each analyze the histogram of visual words and give a probability that the given candidate object is a member of that class. These probabilities are combined under the assumption that each RVM is independent by accumulating them into a multinomial distribution and re-normalizing. The resulting distribution over object classes, along with the geometric details of the object, are sent to the mapper for integration into the model. To establish the effectiveness of the RVM on bag-of-words technique, we performed an experiment on the



(a) Visual words extracted from a candidate object image.

(b) The histogram of visual words appearing in figure 53(a)

Figure 53: Vector quantized SURF visual words. Color indicates which visual word a given feature is assigned to by vector quantization. The size of each circle indicates the scale parameter for the underlying SURF feature.

Caltech 101 dataset [Fei-Fei et al., 2004]. Five categories were selected and objects of interest were extracted using the annotations provided with the dataset. Half of the images were used to train the RVM model using 3-fold cross-validation for RVM radius parameter selection. The resulting RVMs were used to classify images from the other half of the images reserved for the test set. The confusion matrix from this experiment can be seen in figure 54.

4.4 Discussion

We have presented several studies that show how high level objects can be used for landmarks in robot mapping. There are two types of objects that can be found in

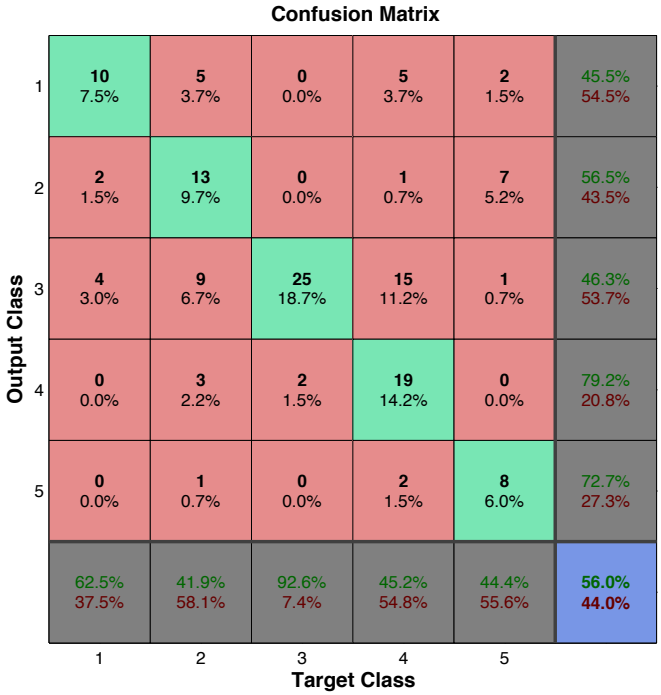


Figure 54: Confusion matrix from a 5 class experiment on the Caltech 101 dataset

houses, permanent and moveable. In a previous chapter 2.3, we described an algorithm which can distinguish these two classes and perform mapping in the presence of moveable landmarks. Combining these two techniques is left for future work.

One of the important characteristics of high level landmarks, like objects, is that they are semantic entities. Previously described feature-based mapping in chapter 2 also considered semantic entities such as walls and doors. The semantics of these landmarks for a mobile robot is simple: you can't drive through a wall, and you can pass through an open door. In this chapter, in section 4.2, we showed how a more semantically meaningful landmark, a door sign, can be used to re-localize a lost robot. Mapping this type of landmark gives rise to a semantic map; a robot could use this landmark information to plan actions which rely on knowing the room numbers in an office building.

The HOG feature classifier used in sections 4.2 and 4.1 did a good job at visually distinguishing door signs from the background. This technique could likely have

been extended for more general object classification (and indeed is the basis of some cutting-edge research along these lines); however, we chose to look at a visual feature based recognition technique. The decision was motivated primarily from the fact that recognition can be achieved with little training data. This comes with the limitation that only textured objects can be recognized.

Objects are important cues in man-made environments because we organize our homes with separate rooms for various purposes, and stock those rooms with objects needed to achieve those purposes. In chapter 5, we will present a novel representation for the semantics which surround objects and rooms. In chapter 6, we will present an algorithm which enables a mobile robot to use this representation to perform object search tasks.

V

PROBABILISTIC COGNITIVE MODEL

Place categorization and object recognition are competencies needed by robots to perform a variety of service tasks in the home, such as fetch-and-carry, retrieval, cleaning, meal preparation, and companionship. Context is a powerful cue for place categorization and object recognition; rooms are laid out in a specific fashion to enable comfortable and efficient living, and objects are used within rooms for tasks specific to that room. This chapter will present a technique which leverages contextual cues for joint reasoning about object and room classification via a conditional random field model.

One of the key contributions of this thesis is the *probabilistic cognitive model* (PCM). It is a collection of software components which allow the robot to reason about the contextual relationships between objects and rooms. This model allows for uncertainty in recognition through probabilistic modeling and uncertainty in position through nonlinear optimization of multiple measurements in SLAM.

We introduced a probabilistic cognitive model for place and object classification using conditional random fields in Rogers III and Christensen [2012]. This chapter is based upon this paper and additional developments.

5.1 Introduction

Domestic service robots will one day work in the home to perform useful tasks such as object retrieval, cleaning and organization, and security. The tireless support of these systems will enable the elderly to live independently by providing service, safety, and companionship. Despite significant automation already being present in the home such as dishwashers, washing machines, and robot vacuums, people face

a steadily increasing amount of duties necessary to support their current lifestyles. In our society, people are looking to spend less time on domestic drudgery, robotic assistance in the household will be a welcomed development.

One of the most important competencies needed for domestic service robots is the ability to understand their surroundings well enough to perform their duties. Robots will be required to interact with objects in people's homes to perform tasks such as cleaning and meal preparation; an understanding of where these objects are located or belong is required to perform these tasks. People prefer to interact with robots in human terms, such as "Get the cup from the kitchen", instead of robot terms, such as "Get object 1372 from (4.2, 12.8, 1.2)". Room category representations in addition to room and object co-occurrence is needed to enable this interaction modality.

A global representation for the location and identity of objects in a domestic environment can be used by a robot to perform a variety of service tasks such as fetch-and-carry, retrieval, cleaning, meal preparation, and companionship. Objects are frequently the point-of-interest for robot missions in the home. People have many unique and distinctive objects in their homes which can be used as landmarks for robot navigation or features for robot mapping.

One of the most important tasks that a new domestic service robot must be capable of is the first one that it will perform when it is unpacked from its shipping crate: mapping its new home and familiarizing itself with the objects with which it will need to interact. The position of an object within the environment can be used as a cue for that object's identity. For example, the microwave oven is more likely to be found in the kitchen, and the toilet is almost exclusively found in the bathroom. The knowledge of the label of the room currently inhabited by the robot can be used to narrow the potential classifications of the objects in the room. The recognition of some objects can also be used as a cue for the identity of other objects around them, such as the mouse is usually to be found to the right of the keyboard, or the light

switch should be found on one side of the doorway.

The task of identifying objects in an unknown (and dynamic) environment should incorporate spatial location and object permanence. In this way, object recognition and SLAM are linked; the performance of each is improved by the other. Object recognition can provide a strong cue for data association, and spatial position can provide a strong cue for object identification. Prior identification of an object from a certain vantage point, combined with *object permanence*, the expectation that things remain where they were last seen for short periods of time, can be used to simplify the future recognition task; it limits the search space and permits less certain matches to be incorporated if they agree with previous measurements of the object.

This chapter will present a technique for combining reasoning about object and room classification within the *OmniMapper* framework described in section 2.4. Object classification consists of two components: first, a direct recognition based on SURF feature matching, and second, a bag-of-words technique for classification of objects which are not recognized by the first component. Room location and extent are measured and placed in a hybrid metric/topological map. Object recognition and classification measurements are provided along with room adjacency and object-in-room relationships to a conditional random field (CRF) model called the *probabilistic cognitive model*, or PCM. The PCM estimates the marginals on each object and room label via loopy belief propagation.

Related work will be presented in section 5.2. The specific algorithms and techniques developed for this chapter will be presented in section 5.3. The experimental procedure will be outlined in section 5.4 and results will be discussed in section 5.5. Conclusions will be presented in section 5.6.

5.2 Related Work

A visual place recognition technique is presented in Ullah et al. [2008]. This technique extracts SIFT features [Lowe, 1999] and performs room recognition with a support vector machine (SVM). This technique is demonstrated to generalize across several experimental settings, and even to images taken by robots with catadioptric as well as perspective cameras. Another technique which uses SIFT features for visual localization is presented in Booij et al. [2007] and a probabilistic model for appearance-based localization or place recognition is presented in Kröse et al. [2001]. Our approach is different from this approach in that we first segment rooms and then recognize objects within that room for classification.

A hybrid metric/topological cognitive model called the Spatial Semantic Hierarchy (SSH) is presented in Kuipers [1998]. This model incorporates representations for robot reactive behaviors and control at its lowest levels. At the higher levels of the hierarchy, the SSH makes use of a topological map representation as well as a metric map representation. Another technique applies the *conceptual spaces* representation in Gärdenfors [2000] to mobile robots by representing topological relationships, metric maps, objects, and people in Zender et al. [2008]. We will also be building a topological map representation to determine room adjacency and which objects appear within each room for use in the CRF model. We also build a metric map of the geometric coordinates of these objects which the robot uses in a simultaneous localization and mapping (SLAM) framework to keep track of the robot's location, as well as to generate the topological map.

A technique for classifying places or rooms based upon range data from a laser scanner is presented in Mozos et al. [2005]. This technique demonstrates good performance in categorizing among three classes of room, corridor, and doorway; however, it is unclear how well this technique would perform at the task of classifying specific types of rooms. The authors extended this work by using this representation for

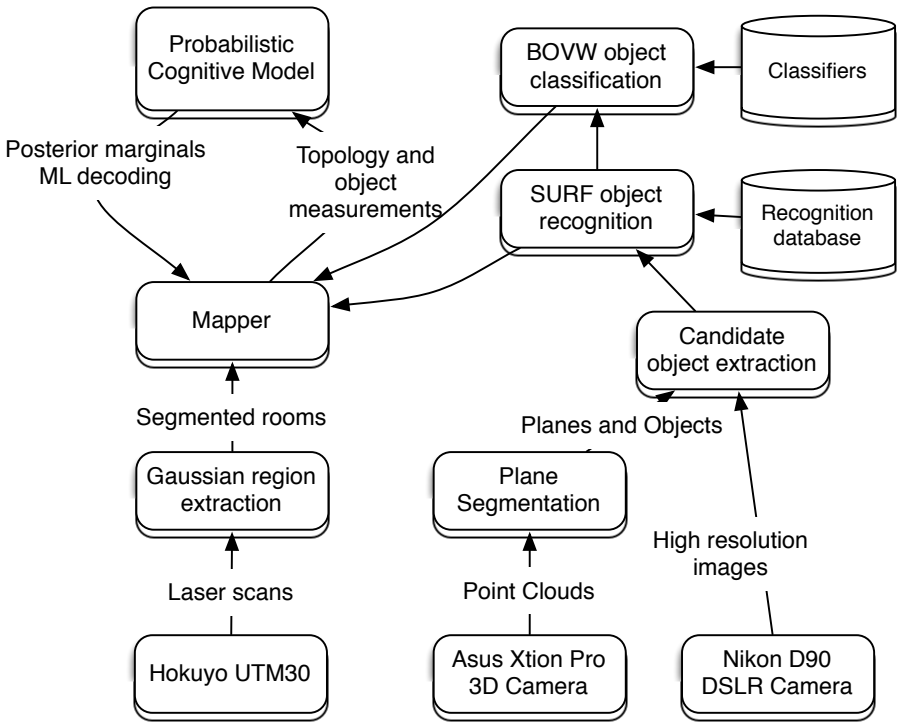


Figure 55: A diagram of the components used in this chapter.

exploration using semantic information in Stachniss et al. [2006].

Context can be a useful cue in recognition. Global features such as bag-of-words based texture recognition can help with scene recognition and improve object recognition [Oliva and Torralba, 2008]. In Shotton et al. [2006], a technique is developed for jointly segmenting and classifying the objects on a per-pixel basis in images. This technique also uses a conditional random field (CRF) model with shape *textons*, color distributions, image locations, and edges to segment and classify objects in the image. Semantic information about object co-occurrence was added to this CRF model in Rabinovich et al. [2007].

5.3 Algorithm

An overview diagram of our system can be seen in figure 55. Laser range measurements are provided by the Hokuyo UTM30 laser scanner to the Gaussian place segmentation module (section 5.3.2), which provides estimates of room shape and size

to the mapping module(section 4.2.4). Point cloud data is gathered by the Asus Xtion Pro depth camera and then is filtered and segmented to extract candidate object point clouds. Candidate object point clouds are then projected into high resolution images from the Nikon D90 camera to extract candidate image regions (section 4.3.1). Object recognition(section 4.3.2) and classification(section 4.3.3) is performed on these candidate image regions and the results are provided to the mapper. The mapper (section 4.2.4) builds a metric map of the location of places and objects, and also builds a topological map which it sends to the probabilistic cognitive model module (section 5.3.1) along with object classification distributions and recognition results. The probabilistic cognitive model module computes the posterior distribution over place and object labels, as well as the maximum likelihood configuration of the world. Detailed descriptions of these modules is provided below.

5.3.1 Probabilistic Cognitive Model

Domestic environments are organized with specific objects located in particular rooms such as toothpaste in the bathroom and calculators in the office. Rooms are also arranged in specific patterns to enable efficient and comfortable living, such as bathrooms are next to bedrooms, and dining rooms are adjacent to kitchens.

The probabilistic cognitive model is a conditional random field (CRF) model of room adjacency and object-room compatibility to reason about these design patterns to determine room label. CRF models express the probability of configurations of variables through a set of compatibility functions. In the case of modeling room adjacency and room-object co-occurrence, there is one type of compatibility function which favors likely pairs of rooms and combinations of rooms and objects.

$$p(y|x) = \frac{1}{Z(x)} \exp \sum_{k=1}^{K} \lambda_k f_k(y, x) \tag{34}$$

In the application of equation 34 used in this chapter, feature functions represent

room adjacency and object in room properties. More appropriate combinations of these values are given larger values by the feature functions.

Conditional random fields (CRF) are a superclass of Markov random fields (MRF), defined as a graph and probability distribution defined as variables Y which are conditioned on observations X:
$p(Y_\nu|X, Y_\omega, \omega \neq \nu) = p(Y_n u|X, Y_\omega, \omega \sim \nu)$ This is the Markov property, with the "$\sim$" operator indicating that a neighbor relationship is present between two variables. Briefly, this property states that the distribution on variables Y when conditioned on X and all other variables is the same as the distribution on Y conditioned on X and a (small) neighborhood of other variables. As in MRFs, CRFs can be represented as the normalized product of exponentials of potential functions: $p(Y_\nu|X, Y_\omega, \omega \sim \nu) = \Pi_{k1} \exp(\lambda_i \psi(y_j, y_l)) \Pi_{k2} \exp(\lambda_i \phi(y_j, x_l))$ In this equation, potential functions ϕ are defined as pairwise potentials between variables, and potential functions ψ are defined as pairwise potentials between observations and variables.

CRFs are discriminatively trained on variables given input measurements. The alternative MRF representation is instead generatively trained on input measurements and variables simultaneously. Generative models require learning or specification of measurement models which brings in more model parameters for estimation, additional distributions to estimate and therefore more reasoning complexity. In comparison, CRFs require no model for measurement states; the only models required are those which represent complementarity of adjacent variables as well as a conditional model of each variable with respect to its input measurement value. This advantage is balanced by an increased complexity in training; CRF training requires the use of a belief propagation algorithm to evaluate gradients, which makes it orders of magnitude slower than MRF training.

The fact that measurement models are not needed in CRF posterior estimation allows for the specification of arbitrary feature functions. In practice, it is necessary

to restrict the scope of these functions to those which involve a small number of variables due to increased complexity in belief propagation.

The CRF model is chosen for this task instead of the more typical Markov Random Field (MRF) model because it allows us to incorporate certain pieces of evidence as absolute, and solve for the distribution of uncertain variables conditioned on this absolute evidence. The evidence which we consider absolute are objects which have been identified by SURF feature matching. The evidence which we consider uncertain are objects *classified* by a bag-of-words classifier. Learning with CRFs can be performed discriminatively, to maximize the likelihood of the labels given the certain evidence. In contrast, learning with MRFs is generative and maximizes the joint likelihood of the labels and evidence.

In our prior work [Rogers III and Christensen, 2012], we used the UGM Matlab package [Schmidt, 2011] to compute the posterior distribution of this model. It is possible to communicate between ROS and Matlab; however, due to some technical difficulty the Matlab implementation and UGM is not used on live robot runs. We have now developed our own loopy belief propagation implementation in C++ which is able to compute the posterior distribution of the model. See appendix 8.2 for a description of the loopy belief propagation algorithm and its implementation.

PCM model parameters fall in two categories: room adjacency and object-in-room affinity. These parameters were trained separately. In our prior work [Rogers III and Christensen, 2012], model parameters were selected by hand. The first type of model parameter is the room adjacency affinity. The training data for the room adjacency model was generated by manually extracting the topology of a set of single-story house floor plans[1]. The floor-plan topology was represented as an adjacency matrix with a 1 in entry (i, j) if room i is topologically connected (adjacent) to room j.

[1]House floor plans were extracted from the book "Lowe's 1-story home plans". Creative Homeowner, 2007.

	Kit	Hall	LR	Bath	Bed	Office
Kitchen	0.82	0.84	2.14	0.67	0.84	0.77
Hall	0.84	0.65	1.87	1.55	3.25	1.58
LivingRoom	2.14	1.87	0.76	0.63	0.57	0.72
Bathroom	0.67	1.55	0.63	0.75	3.30	0.72
Bedroom	0.84	3.25	0.57	3.30	0.65	0.66
Office	0.77	1.58	0.72	0.72	0.66	0.89

Table 12: The room adjacency model trained from floor plan topologies. This model was trained using a modified version of the training routine in UGM [Schmidt, 2011] to use heterogenous graph topologies.

Additionally, ground truth labels are provided for each room. Since architects do not usually specify which bedroom or space is to be used as the *office*, we selected one of the bedrooms in plans with three or more to be an *office*.

The L-BFGS model parameter training procedure implemented in UGM was used to optimize adjacency affinity parameters. The L-BFGS method makes an approximation to the Hessian to achieve quadratic convergence with Newton's method. Model parameters are optimized to minimize the negative log-likelihood of the training data. This training procedure is typically run on a fixed graph topology with many data instances; however, in our case we have heterogenous graph topologies which share model parameters. The training procedure was modified to work with heterogenous graph structures by computing the sum of the negative log likelihood and the sum of the gradients across all graph topologies. Additionally, a regularization term was added to the optimization to penalize large parameter values. The resulting room adjacency model is shown in table 12. In this table, common adjacency relationships are given a higher affinity. For example, *bedroom* and *bathroom* have the highest affinity. Additionally, *living room* and *kitchen* have a large affinity and are likely to be found adjacent to one another.

The object-in-room affinity model was developed by analyzing sentences in the

Open Mind Indoor Commonsense Database [Kochenderfer and Gupta, 2004][2]. This database consists of sentences describing relationships between entities which are indoors, in an office or domestic environment. A series of queries was performed to find the number of sentences in the database which contained both a place label from the set (*kitchen*, *living room*, *bathroom*, *bedroom*, *office*) together with the object class labels and a group of instance examples for each class label. Projects like *ConceptNet* look for predicates and use these to associate words. The assumption made for our training model is that if a place and an object are mentioned in a sentence, then they are related (and therefore this object might be found in this room). The relative frequency of sentences for an object class is used to construct a model with respect to room label. Combinations with no representation in the set are assumed to have a minimum count of one. These models are shown in figure 56.

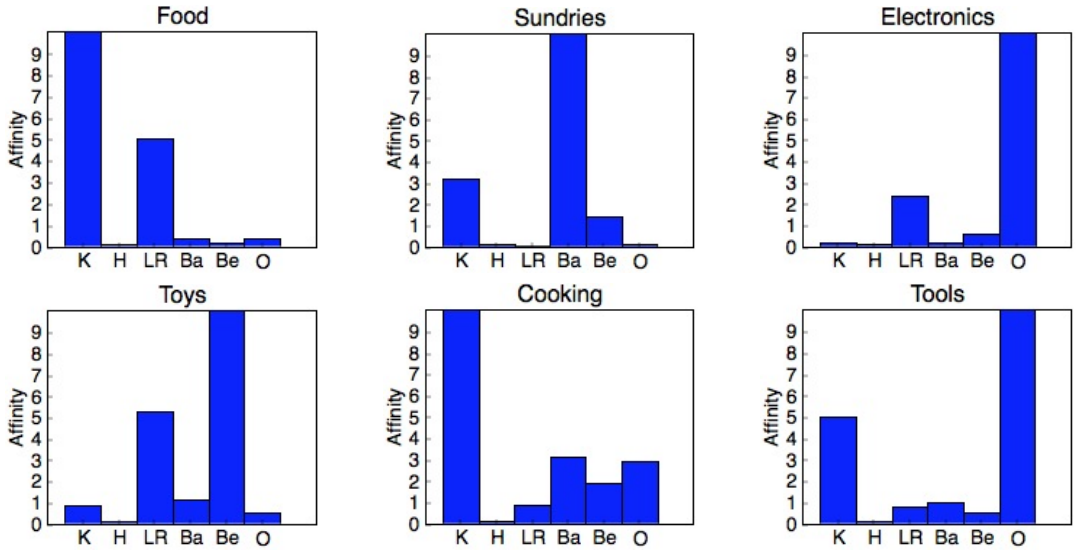


Figure 56: Object to room affinity models determined from Open Mind Indoor Common Sense database. Axis labels are abbreviated as K: Kitchen, H: Hall, LR: Living Room, Ba: Bathroom, Be: Bedroom, O: Office.

We use the technique described in a previous chapter, in section 4.3.1 to segment objects out of point-clouds by finding point-cloud clusters protruding above tabletop

[2]OMICS is based on Concept Net and can be found at http://openmind.hri-us.com

surfaces. Object recognition is performed by the algorithm described in section 4.3.2. If the object cannot be recognized, then it is classified by the module described in section 4.3.3.

5.3.2 Gaussian Place Segmentation

The OmniMapper is designed to create metric maps of generic landmarks, and it is used here to map where objects appear in the environment and localize the robot with respect to these objects. A topological map which describes room regions and their connectivity is needed to identify the room nodes and the edges between them for the graphical model described in this chapter. The technique used by our robot to build this topological map is to leverage the *Gaussian regional analysis* technique described in Neito-Granda et al. [2010], where laser scan data is analyzed to find a Gaussian ellipsoidal region approximation to the space. The metric mapper from section 2.4 identifies links metric robot trajectory elements with these Gaussian ellipsoidal regions. When the Mahalanobis distance to the current ellipsoidal region is above a threshold, the robot adds a new region and an edge link to the prior region. The mapper also checks the distance to each Gaussian region and transitions into the one which is closest in Mahalanobis sense. A new edge link is also added to the graph if one was not present between these two rooms, since the robot has just found a path between them. A thick blue line represents the topological link between these two rooms. When object recognitions or classification distributions are observed by the robot, they are assigned to the room Gaussian where the robot currently resides. Each room Gaussian is given a distribution over room labels as a result of the PCM described in section 5.3.1. The Gaussian ellipsoidal regions can be seen in figure 59(a); the green ellipse is the robot's current room, and the yellow one is the prior room.

Figure 57: Our mobile robot "Jeeves", inspecting a cup on a table in our test environment.

5.4 Experiment

The robot experiments in this chapter are performed on our mobile robot, Jeeves. Jeeves is a Segway RMP200 base which has been modified to be statically stable with support wheels, which can be seen in figure 57. The robot uses an Asus Xtion Pro 3D camera for object segmentation. The robot has several laser scanners which are used for obstacle avoidance, localization, mapping, and measuring the size of rooms; however, in this application the robot is tele-operated and uses the Hokuyo UTM30 laser scanner to measure rooms. The robot uses a Directed Perception PTU-46-70 pan-tilt unit to aim sensors on its upper extremity. Currently, the only sensor on the pan-tilt unit is a Nikon D90 DSLR camera with an 18mm lens. This high resolution camera is used to gather detailed images of target objects for classification and recognition. Our group uses the Robot Operating System (ROS) developed at Willow Garage [Quigley et al., 2009] for low level behavior and interprocess communication.

We performed a series of experiments on log data gathered from the Aware Home test facility at Georgia Tech. We believe that the topology of the Aware Home is typical of modern architecture. There is a kitchen with a small dining area opens into the living room. The living room is connected to a hallway which connects to an

office, bathroom, two bedrooms, and a closet. At the time of these experiments, the Asus Xtion Pro 3D camera was placed on the robot at a height that makes it difficult to observe tables and objects if they are more than about 1 meter above the ground, so we made sure that there was a counter, desk, or table at or below this height. The 3D camera has subsequently been moved to the pan-tilt unit alongside the high resolution camera to enable observations of surfaces and objects at any reasonable height.

We tele-operated the robot in a set of trajectories in the kitchen, living room, office and bathroom. In each room, the robot was made to collect high resolution camera images of the table surfaces and the objects thereupon.

When an image is captured by the high resolution camera, the 3D camera captures a point cloud. The point cloud is filtered to select a volume of interest in front of the robot. Planes are extracted from the remaining points, and objects are selected as clusters of points which lie above horizontal planes. The object points are projected into the high resolution image to find a region of interest for visual feature analysis. Each region of interest becomes a candidate object image.

Some examples of trained objects and the categories to which they belong can be seen in figure 58(a). The current set of object classes include *food*, *toys*, *cooking*, *sundries* (medicine, soap, etc.), *tools*, and *electronics*. This taxonomy was selected to assign objects to classes which should logically correspond to room assignment. The rooms currently available to the PCM are *kitchen*, *bathroom*, *living room*, *office*, *hall*, and *bedroom*.

5.5 Results

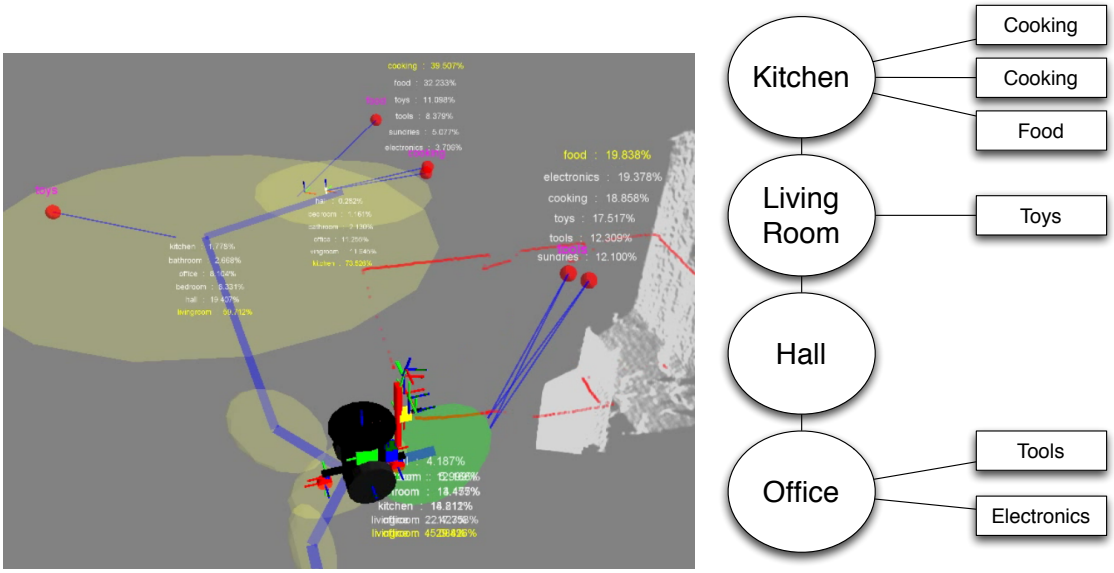
The posterior marginal distributions over some of the objects and rooms in a test run can be seen in figure 59(a). The most likely decoding can be seen in figure 59(b).

In one test run, the first time the robot entered the office it failed to recognize the

(a) The six classes of objects with example images.

(b) Entire recognition database.

Figure 58: Object classes and instances which can be used by the PCM



(a) Room nodes are shown as yellow ellipsoids, except the current room is shown in green. Topological room connectivity is shown with thick blue lines between room nodes. Posterior room label distributions is shown below each room node (unlabeled nodes are Hall nodes, and are omitted for clarity). The yellow entry is the most likely marginal room label. Objects are shown as red spheres with thin blue lines linking them to the poses from which they are observed. Recognized objects have their labels displayed in pink.

(b) The most likely configuration of rooms and places from a test run in the Aware Home (redrawn, some extraneous Hall nodes removed for clarity).

Figure 59: Output posterior marginal distributions and graphical display shown in figure 59(a). Most likely configuration (decoding) shown in figure 59(b)

Dremel tool or the Mac mini with SURF feature matching. The Mac mini was not part of the recognition database, and it was mistakenly categorized as a Food. This resulted in the office posterior distribution favoring the label "living room". When the robot re-entered the office at a later point in the test run, the Dremel tool was recognized directly and the office was relabeled correctly.

The Gaussian regions used for place segmentation performed poorly due to the fact that the laser scanner could only observe a 180 degree arc in front of the robot. When the robot is driven in reverse, it quickly exits the current Gaussian region and creates a new one. This new region similarly favors the region in front of the robot and therefore is also quickly exited. This problem was addressed by looking at the current Gaussian region identified by the robot. If the current Gaussian region largely overlaps a previous region, then the regions are not updated even if the robot's pose lies outside the current set of Gaussian regions at a given Mahalanobis distance threshold. A side effect of this change is that the Gaussian regions will now be non-overlapping in the topological map.

5.6 Discussion

We have demonstrated a model which is used to jointly classify objects and room labels on a mobile robot. This model incorporates information from two types of object measurements: recognition of previously seen and trained objects, as well as classification of novel objects. This model was tested on a mobile robot tele-operated in a real domestic environment with objects. The current experiments focus on determining that these recognition, mapping, and reasoning components can be made to work together to accomplish rudimentary understanding of the purpose and structure of a domestic environment. The selection of objects used in the experimental tests consisted of many of the same objects which were explicitly used for training in the recognition and classification components; however, we maintain that it is

reasonable and even desirable for a mobile robot to acquire models of the objects with which it will be interacting through human interaction in addition to self training.

In the absence of the influence of object and place contextual relationships, i.e. without the use of the PCM or with a PCM given uniform uninformative contextual models, the object recognition and classification system would have been able to function with degraded performance compared to the results presented in this chapter. The recognition module would have performed the same; when it identifies an object which is present in the database, it is recognized absolutely. Objects which are not recognized have a categorical distribution over their labels. In the absence of the context of their surroundings, as provided by the PCM, these categorical distributions over labels are only based upon the output of the classification module. With the PCM, these categorical distributions over labels are computed as the product of two factors. The first factor is the same without the PCM; it is the result from the classification module. The second factor contains the contextual information carried by the PCM. These two factors taken together generate the posterior marginal distribution over the object. This result contains not only what the classification module knows about objects, but also what is known about the context of this object in its environment.

In addition, objects which have been *glimpsed* by the system but have not yet been analyzed for recognition or classification, benefit from context through the PCM. This aspect is less relevant in this chapter, where the robot is tele-operated and is made to take pictures of all interesting objects. In the next chapter, the robot will need to choose which objects to analyze to find what it is looking for. In this situation, the robot will make use of the contextual relationship of other objects and places on an unknown object to evaluate its label distribution before it is observed in more detail.

The intended application of the PCM is to enable autonomous exploration and mapping, as well as service tasks. The current implementation establishes that this

reasoning model is useful for leveraging context for classification of objects and places. The posterior marginal distribution on the PCM can be used to direct exploration. The robot can search for additional objects in rooms where the label entropy is large, or it can examine unknown objects in greater detail. The robot can select certain unknown objects to pose a small set of questions to a human operator by selecting the objects which are most likely to provide a great deal of information to the rest of the model when they become disambiguated.

Other tasks which use the PCM to understand human commands should use the maximum likelihood decoding (which is not in general the same as the maximum marginal likelihood) of the entire graph. In this way, the robot can use the context of the entire model to perform tasks such as "get a cup from the kitchen".

Model parameters are trained from data before operation. New experiences are not currently used to improve robot performance. This training information should also be augmented by online adaptation to new observations.

The current recognition database is limited in size due to the amount of time needed to perform SURF feature matching on high resolution images. We will expand object classes and recognition database to cover the key components which will be useful to robots in performing their tasks as well as in understanding their environments.

To better account for actual module performance and the circumstances of the robot's actual deployment, a life-long training method should be developed. The training techniques described above assumed that the object models and room adjacency models could be decomposed and trained separately. These model parameters are joined into a final set of model parameters. This method seems to be a good way to initially get good values for model parameters, but does not incorporate any representation for unexpected actual values, such as over-segmentation by the place segmentation model or errors in the object recognition module.

When the robot has completed exploration in an unknown environment, either in simulation or in actual operation, the operator would be given the opportunity to provide ground-truth place labels. Model parameters could then be updated via stochastic gradient descent (SGD) to match this new training example. This training is currently implemented offline to leverage conditional random field parameter estimation techniques offered by the UGM library in Matlab; however, it could be adapted to provide online training.

In the next chapter, 6, we will describe how the PCM can be used to provide high-level control for a mobile robot. This model is used in two applications to select actions to perform an object search task in a context-aware manner.

VI

PROBABILISTIC COGNITIVE MODEL PLANNER

Context is an important factor for domestic service robots to consider when interpreting their environments to perform tasks. In people's homes, rooms are laid out in a specific arrangement to enable comfortable and efficient living; for example, the living room is central to the house, and the dining room is adjacent to the kitchen. The identity of the objects in a room are a strong cue for determining that room's purpose. This chapter will present a planner based upon the *probabilistic cognitive model* (PCM) from chapter 5 which enables an autonomous mobile robot system to use room connectivity topology and object understanding as context for an object search task in a domestic environment.

6.1 Introduction

Domestic service robots can enrich our lives by performing chores in the home such as meal preparation, fetch and carry tasks, security, clean-up, and companionship. Not only will robots in the home make people's lives more convenient, they will also enable the elderly to stay in their homes instead of in an expensive nursing care facility.

Context is an important cue which can be leveraged to improve the performance of robots in a variety of domestic tasks. For example, a service robot is asked to fetch a snack, but it doesn't know where the snacks are, because the human didn't put them away last time. Where should the robot search first for the snack? With no understanding of the semantics of context, the robot will waste its time searching the bathroom when it should be focusing its search in the kitchen, office, and living room.

Many domestic service robot tasks will require the ability to recognize and locate

a significant fraction of the objects in their user's home. If the robot is asked to load the dishwasher, then it must first know where to search for the dishes. Some dishes will be found in the kitchen; however, the robot should not ignore the rest of the house or it will risk failing to perform its duties most efficiently. A complete and exhaustive search of every room is too time consuming for the robot's busy schedule of duties. A robot must understand the context of where objects should be found in the home to perform object retrieval, meal preparation, and clean-up tasks, to enable it to focus its search where task-relevant objects are most likely to be found.

Context can be represented as a graph connecting related places together with objects which can be found in those places. This type of graph can be used in a bottom up fashion to infer room labels from object recognition [Rogers III and Christensen, 2012]. Conversely, this graph can also be used in a top-down fashion to predict the presence (or absence) of objects based upon room categories. This type of reasoning will be combined with a planner to select actions on a mobile robot to perform an object search task in this chapter.

This chapter will describe a novel technique for combining the probabilistic model described in Rogers III and Christensen [2012] for context with a high-level robot action planner to accomplish an object search task in a domestic environment. The planner will be shown to outperform an uninformed search strategy in a series of experiments performed in a simulation. Additional robot experiments are performed with our prototype domestic service robot in a domestic setting, and a military robot in a military training scenario. These live robot experiments demonstrate the advantage of using the PCM-Planner to perform an object search task which is informed by context.

We present related work in section 6.2, and an outline of the algorithms in section 6.3. Three experimental scenarios are presented in section 6.5. This chapter closes with a discussion of the key results in section 6.6.

6.2 Related Work

Recent approaches to indoor scene recognition based upon semantic understanding of objects and places have emerged. In Espinace et al. [2010], the authors demonstrate a technique where category level visual object detection is used with a hierarchical probabilistic model to perform visual scene categorization. The authors use a boosted cascade of several feature types for category level object detection. They use a naïve Bayes model to give a distribution over scene label given the output of the object categorizers.

The use of context in predicting where to find objects was shown in Kollar and Roy [2009]. In this work, a co-occurrence database of object names is built from WordNet and Flikr. This paper demonstrates an algorithm where a query object can be predicted based upon the knowledge of other objects in the scene. The authors also describe a planning algorithm which selects paths for a robot which search for the query object using their model.

Active visual search is the problem of using context to improve the performance of an object search task. Recent work in Aydemir et al. [2011], present a technique were an object search is performed on a mobile robot using probabilistic reasoning about object/place co-occurrence. This technique uses a planner which hypothesizes *virtual objects and rooms* which potentially help the robot accomplish its visual search task.

The partially-observable Markov decision process (POMDP) has been used for high-level robot control. Since brute-force value iteration is intractable, Pineau and Thrun [2002] looked at exploiting sparsity and the sequential dependencies from a set of actions to solve a simpler problem. Another approach [Roy et al., 2005] has mitigated the complexity of using POMDPs for mobile robot control through the use of compression to compute approximately optimal policies.

In this chapter, we extend the idea of using *virtual objects and rooms* of Aydemir et al. [2011] through the use of our probabilistic model. We leverage the context of

places seen in a topological map when considering which unknown region to explore, to be the one most likely to contain the object for which the robot is searching. Each object found in a room will influence the label on the room and affect the likelihood of finding the target object.

6.3 Algorithm

Robots operating in the real world must deal with uncertainty and errors in measurement and actuation. A probabilistic representation can be used to acknowledge this uncertainty and reason about it. Our probabilistic representation for the robot's *belief state* about its environment is called the *probabilistic cognitive model*, or PCM. This model was described in chapter 5. Briefly, the PCM is a *probabilistic graphical model* relating various entities with which the robot is expected to interact. This model allows the establishment of contextual relationships between these entities. The nature of these contextual relationships could take many different forms; however, for this work we have investigated the *within* relationship between objects and places, and the *adjacent* relationship between places. This model has been trained with data from house floor-plans and from the Open Mind Indoor Common Sense database.

The method for using the PCM for autonomous mobile robot control is described in section 6.3.1. This method uses the PCM to represent the continuous state space of a Markov decision process, and to estimate the effects of the robot's actions to select actions which are likely to achieve the robot's goals. Some additional components were developed to enable operation in an unknown environment. Frontier-based exploration is described in section 6.3.2; this technique is used to hypothesize that additional spaces or rooms might need to be explored. Since the robot has a finite database of object models, we developed a user interface through which additional models could be trained of unrecognized objects; this interface is described in section 6.3.3. These methods will then be demonstrated in simulation and real

experiments in a variety of settings in section 6.5.

6.3.1 Probabilistic Cognitive Model Planner

Our contribution for this chapter is a planning module which uses the PCM described in chapter 5 to select robot actions to perform an object search task. Briefly described, the PCM Planner hypothesizes the effects of its selected actions on the PCM and chooses the action sequence which is most likely to find the object of interest.

The PCM Planner performs a depth-first-search on a PCM by trying potential actions at each level and hypothesizing new PCM states based on the result of these actions. For these experiments, the set of robot actions is: *Move*, *Search*, *Examine*, and *Fetch*. The marginals in the PCM are used to evaluate the probability of certain conditions which are used to assign rewards for performing actions. The reward value (given in table 13) is multiplied by this marginal to get the predicted reward value.

Reward values are specified by hand at this time and can be seen in table 13. The reward values for the Fetch action are tuned relative to one another; the penalty for fetching the wrong object is much larger than the reward for fetching the correct object. Because of this disparity, the robot will need to be quite sure that an object is the correct one before it retrieves it. The rewards for the other actions are small negative values which are roughly proportional to the amount of effort and time necessary to execute these actions. Other values of rewards could be specified to favor certain actions. An optimal set of reward values could also be learned with respect to a certain task; however, this is reserved for future work.

The *Move* action can be performed to attempt to transition to another room which is topologically adjacent to the one in which the robot currently inhabits. The reward function for this action is currently a small negative constant amount for each edge traversed on the topological map. This action is necessary because the scope of the remaining actions is limited to occur only within the room in which the robot

Action	Reward
Move	-3
Search	-5
Examine	-4
Fetch(correct)	20
Fetch(incorrect)	-100

Table 13: The rewards assigned for performing each action

currently resides.

The *Search* action is used to try to find objects in the current room using the segmentation module described in section 4.3.1. The result of the *search* action in the hypothesized next state of the PCM is that a new object is found by the robot. A uniform prior is placed on this new object; after the posterior is computed with loopy belief propagation, the context of the current room will affect the label of the hypothesized object.

The *Examine* action is selected by the planner to look at a previously segmented or categorized object which has not been recognized, and try to identify it directly with the recognition module described in section 4.3.2. The hypothesized next PCM state upgrades the object categorized label distribution to a clamped, known recognition.

The final action, *Fetch*, is terminal and represents the robot making its final choice; choosing this object as the solution to the object search task.

The PCM planner chooses actions in a partially-observable Markov decision process (POMDP). In a POMDP, the system state is represented as a belief state, which is a probability distribution over states. In our application, the belief state is represented by the PCM. The specification of a POMDP consists of a representation for the system state, an action model which describes how the belief state is updated as a result of actions, an observation function, and a reward function. In this application, the reward function provides a positive reward for performing the Fetch action on the desired object, and a strong negative reward for performing the Fetch action on the

wrong object, as seen in table 13. The robot also receives a small negative reward for performing each of the non-terminal actions to prevent loops or overly conservative behavior. The state transition function is given by the rules for adding hypothesized elements to the PCM as described above. In our implementation, observation actions are implemented as state transition functions.

The PCM planner computes a sequence of actions leading it to the terminal *Fetch* action; however, since this relies heavily upon hypothesized results, only the first action in the sequence is executed before re-planning. A future improvement would be to detect that the new state is equivalent to the predicted outcome and continue executing the plan, without re-planning. The *Move* action sends coordinates of the center of the target room Gaussian as the destination for the motion controller. The *Search* action instructs the base controller to move to waypoints aligned with the major and minor axes of ellipsoidal regions described by the covariance matrix of the room Gaussian. While the robot is moving to these positions, objects on table surfaces are likely to be seen as they pass nearby. The *Examine* action moves the base to within 2 meters of the target object, saccades the PTU to aim the DSLR camera, and takes a high resolution image which is processed by the recognition and classification modules. The PCM is updated asynchronously and its current state will be used by the PCM planner during the next planning interval.

The PCM Planner expands the search tree of actions until it reaches a terminal state, where expected rewards are computed. The sequence of actions which results in the highest expected reward is then returned, and the first action on that sequence is performed. Consider a sequence of actions where the robot has a starting state for the PCM where it has just observed an object which indicates that it is in the Kitchen, but it is looking for the TV remote, which is most likely to be found in the Living Room, which is probably adjacent to the Kitchen.

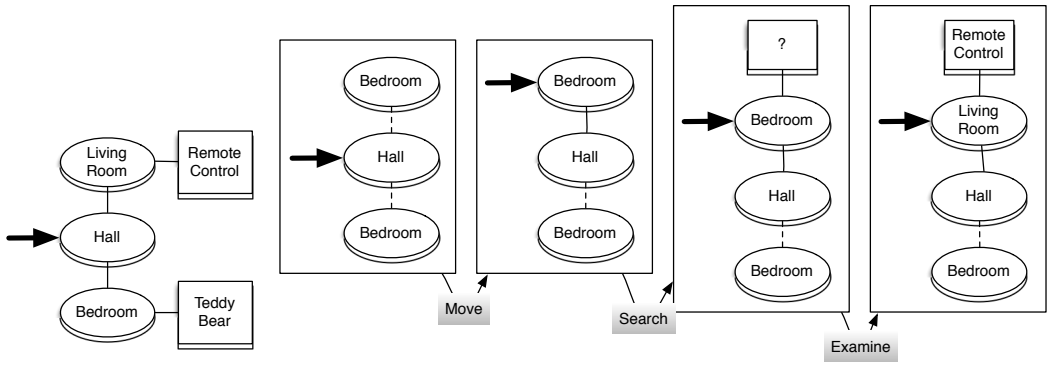
The planner will evaluate many possible sequences of actions to find the best one.

First, the planner will consider *Search*ing for more objects in the room; if it finds one then it will *Examine* it and hopefully *Fetch* it if it turns out to be the right object.

The reward for this sequence of actions can be computed by summing the expected reward for each action along the sequence with the distribution given by the hypothesized PCM. The first action, *Search*, always gives a reward of -5[1]. The hypothesized PCM after the Search is performed will contain an additional object in this room. The second action, *Examine*, has a reward of -4. When this action is performed, the hypothesized PCM has upgraded the new object to a clamped recognition, but it keeps track of the marginal on this object. This marginal is generated purely by the context of the location of this hypothesized object; the fact that it is in this room gives its marginal. Let p be the probability that the new object is from the goal class. The final action, *Fetch* will then generate the reward $p * 20 + (1 - p) * (-100)$. So the total reward for this sequence is $-5 - 4 + p * 20 - (1 - p) * 100$. If p is small, in a situation where the room we are searching is unlikely to contain the goal object class, this reward will be small. In this case, moving to an unexplored adjacent room first would generate a much higher reward, especially since the goal object class is likely to be found in a room next to the Kitchen.

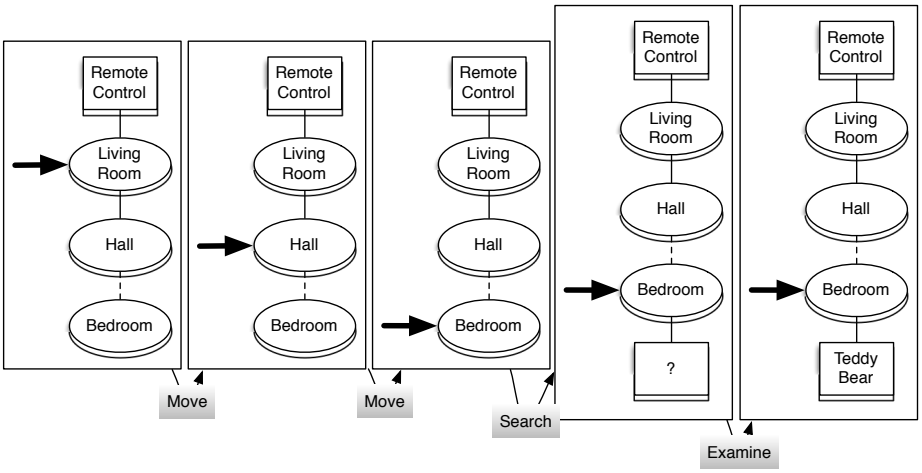
A hypothetical example of the PCM-Planner in operation can be seen in figure 60. In this example, the robot is searching for a "Teddy Bear", which is likely to be found in the Bedroom. The robot starts in the hallway and is aware of two rooms attached to this room. The PCM state is updated as the robot incorporates new actions, but it can predict the effect of its actions on the PCM. The first plan sequence in this example would be {Move, Search, Examine, Fetch}. This sequence is shown being executed in figure 60(b). Once the robot examines the object that it finds in the top room, it recognizes that it is not a "Teddy Bear", so the robot re-plans and

[1]In the actual implementation, the reward for Search is made progressively worse each time it is performed in a given room. A similar mechanism is used to make repeated Examine actions have reduced rewards



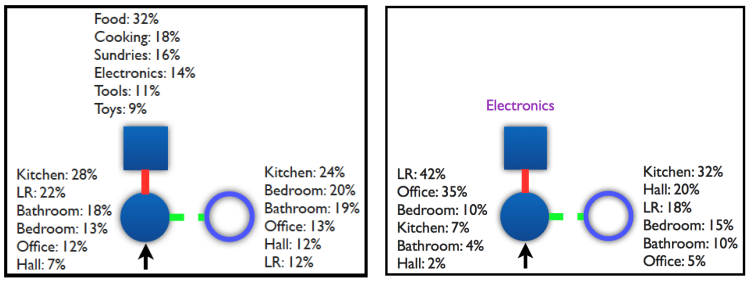
(a) Ground Truth

(b) Initial PCM. Robot searches top room first.



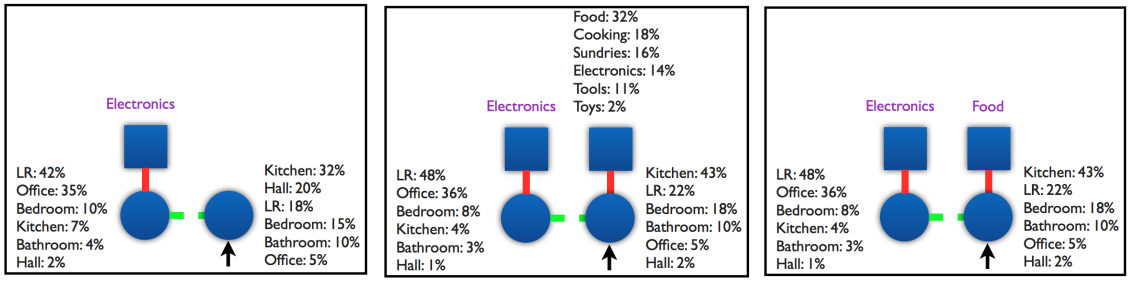
(c) After the PCM state incorporates new information, the robot re-plans: now it searches the lower room. Here it finds the target object.

Figure 60: An example sequence of actions that could be executed by a robot using the PCM-Planner to search for a Teddy Bear. From the initial state, the robot chooses to search the upper room first. When an object is examined in this room, it is found to be a Remote Control. The PCM state is updated with this new information and the robot chooses to move to the lower room and search there. Here, the robot finds the target object. Labels shown indicate the most likely value for each room, in practice these values are given by a probability distribution. A simpler example is shown in figure 61 in greater detail.



(a) Initial state

(b) The object is examined.



(c) The robot moves to another room

(d) The robot searches and finds another object

(e) The new object is examined

Figure 61: A simple world with two rooms and two objects. Each subfigure depicts a PCM state as a hypothetical robot explores this simple world. Unknown quantities are given by probability distributions over labels. Recognized objects are denoted by a single label. A black arrow indicates the location of the robot.

executes the sequence {Move, Move, Search, Examine, Fetch}, which can be seen in figure 60(c). Here, the robot finds the object that it is searching for.

Another hypothetical example search sequence with a simpler environment is shown in greater detail in figure 61. In this example, the probability distributions on unknown objects and places is shown. Each subfigure is one PCM state. The first plan sequence tried by the robot is {Examine, Fetch}. Once the robot examines the first object, it must re-plan since it is looking for Food instead of Electronics. The second plan sequence is {Move, Search, Examine, Fetch}. This sequence results in finding a new object, followed by recognizing it. This is the correct object, so it can be fetched.

The specific function which is optimized by the PCM-Planner is:

$$E[R(s_0, \{a_i\})] = \sum_i p(a_i, s) E[R(s_i, a_i)] \tag{35}$$

$$s_i = T(s_{i-1}, a_{i-1}) \tag{36}$$

In this equation, the robot gets a reward for performing an action in a state given by function $R(s_i, a_i)$, and shown in table 13. In some cases, such as fetching an object, the reward function takes multiple values based upon the state. In this case, the expectation of the reward can be computed by analyzing the marginals on the relevant object being fetched. The reward value can be computed for this action as:

$$E[R(s_i, FETCH(k))] = p(k == t) * F^+ + (1 - p(k == t)) * F^- \tag{37}$$

Here, F^+ is the reward for fetching the correct object, which is 20, and F^- is the reward for fetching the wrong object, which is -100. The planner searches over all sequences of actions and selects the sequence which maximizes equation 35.

Computing optimal policies for POMDPs is intractable in general; but we have exploited sparsity and action dependency to simplify the policy search process, as in Pineau and Thrun [2002]. The sparsity property allows the search procedure to visit a relatively small number of possible belief states when selecting the next action.

Figure 62: The interface by which a user can provide instance and class labels for unknown objects. The robot then adds this new object to its recognition system.

In addition, the action sequence features a hierarchical dependency which further reduces the complexity of the search space.

6.3.2 Frontier Based Exploration

We employ a frontier-based exploration algorithm similar to the one described in Yamauchi [1997] to find potential unexplored rooms adjacent to known rooms. This algorithm examines a costmap generated from laser scan data. Costmap cells have three possible values: Occupied, Clear, and Unknown. We identify the frontier for exploration as the boundary between Unknown and Clear cells; this boundary is the unknown space adjacent to costmap cells that have been seen through. These cells on the boundary are clustered. If the clusters of cells are large enough, then they are seen as *hypothetical rooms* by the planning system which can decide to explore them to search for the target object.

6.3.3 Online model training

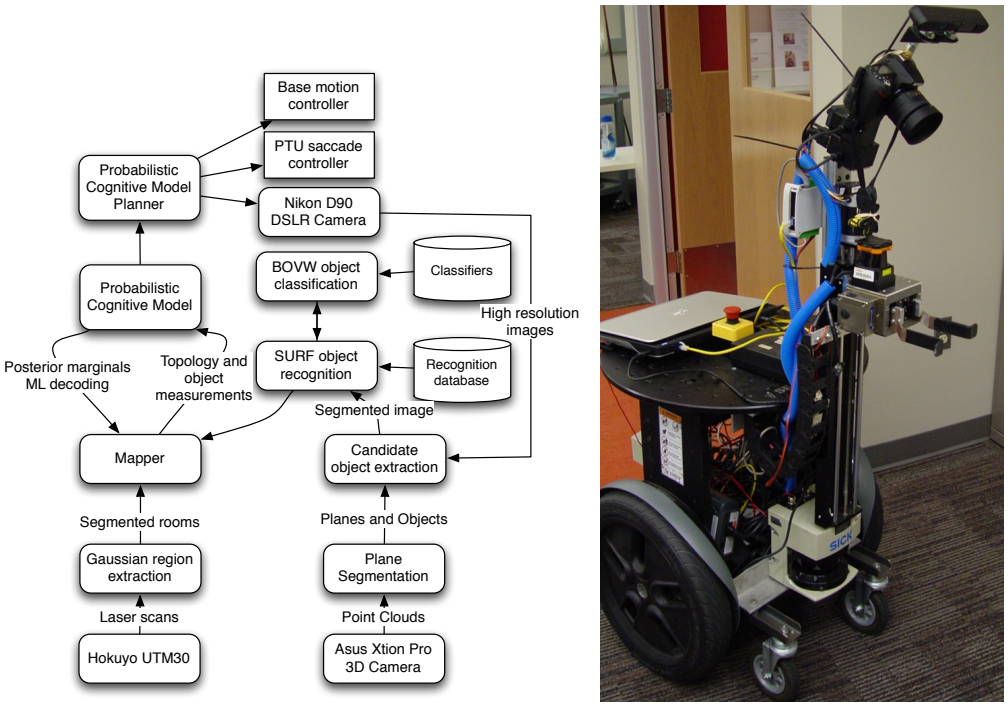
If an object (or canonical view) is not recognized by the current database, then a user is provided with an opportunity to provide a class and instance label to incorporate

it into the database. The user interface is shown in figure 62. This object model will then be available for the robot to use in current and future operation. Additionally, classification modules can be re-trained and updated for online use; however, this is currently performed off-line for use by the robot in its next operation.

Since the robot is able to find objects autonomously through the use of the object segmentation module described in section 4.3.1, this segmented region-of-interest is available to present to a user for identification. There are many potential ways this human assistance could be given. First, an automated image-based search using GoogleGoggles could be made (and we have experimented with this for reading text on signs, see section 4.2.3.1). Another mechanism could be a call center of employees or contractors of the robot's manufacturer who are ready to help the robot identify and understand unexpected objects. This type of training could also be *crowd-sourced* through something like Amazon Mechanical Turk; the robot would pay a small amount to get this training. Finally, the robot could simply ask the user for a label.

6.4 Implementation

The first mobile robot used in this chapter is called Jeeves, see figure 63(b). A software data flow diagram can be seen in figure 63(a). Jeeves is a Segway RMP200 which has been rendered statically stable through the use of caster wheels. Jeeves uses a Mac Mini and a Dell laptop which are networked together to run all algorithms used for these experiments. The robot uses a Hokuyo UTM30 for obstacle avoidance and region segmentation, see section 5.3.2. The robot is equipped with a Schunk PG-60 gripper and a linear actuator; these will be used for object retrieval in future work. The robot uses a Nikon D90 DSLR camera with a 35mm lens in conjunction with an Asus Xtion Pro Live 3D camera for object segmentation, recognition, and classification. The DSLR and Asus Xtion Pro Live cameras are mounted on a Directed



(a) A diagram of the components used for the PCM Planner

(b) Jeeves: The Mobile Robot Valet

Figure 63: Software and hardware components used in PCM Planner experiments

Perception PTU-46-70 pan-tilt unit (PTU) for fine saccade control; the robot uses this to aim the cameras at target objects. Additionally, the PTU is actuated in a nodding search pattern when the robot is exploring to expand the field-of-view of the sensors.

The software used in our mobile robot is built on the Robot Operating System (ROS) [Quigley et al., 2009] for interprocess communication, sensor data integration, and platform control. We use the GTsam nonlinear optimization library with a modular mapping framework that we have developed called OmniMapper. Room regions are segmented using a Gaussian region analysis technique described in section 5.3.2 and in Neito-Granda et al. [2010]. The robot examines objects with 3D point cloud analysis, as shown in section 4.3.1. Object recognition is performed by matching geometrically consistent SURF features in section 4.3.2. Object classification takes place if the object is not recognized directly. The classification technique is described

in section 4.3.3; it is a group of relevance vector machines trained on visual word histograms quantized from SURF features.

The location of the robot, objects, and rooms is determined via the *OmniMapper* described in section 2.4. This application makes use of the *canonical scan matching* plugin from section 2.4.9 and the object mapping plugin from section 2.4.6. The *canonical scan matching* plugin builds an occupancy grid cost map which is analyzed for exploration frontiers, as well as for traversability for motion planners. The object mapping plugin keeps track of the location of objects, and it also performs smart data association and manages the data coming from segmentation and recognition modules.

6.5 Experiments

First, a simulated domestic service scenario is presented in section 6.5.1 which establishes the advantage of using context over an uninformed strategy. The model used in the simulation is then run on our prototype domestic service robot in section 6.5.2 in a home scenario. Finally, a military counter-insurgency scenario is presented in section 6.5.3 where a military robot is used to explore a potential insurgent holdout while looking for weapons of mass destruction.

6.5.1 Simulated experiments

The PCM Planner algorithm was first tested in a simulation environment which abstracted away the performance of the perception components to demonstrate effectiveness. In the simulation scenario, object segmentation, classification, and recognition are replaced with an object simulator which responds to control commands from the PCM Planner to mimic ideal robot performance. The use of a simulation allows for more focused analysis of the relevant algorithms with statistical significance.

Robot software components are implemented in the ROS framework. This framework is built on a set of distributed programs which communicate over *topics*, so

modules interact via defined interfaces. The physical hardware components of the mobile robot, such as the Segway and laser sensors, are simulated in the *Stage* environment. The experimental setting consists of a 2D floor-plan and a set of objects. For each floor-plan used in the experiment, a polygon is defined for each room via a simplicial complex, via a set of line equations.

Objects are sampled for each room for a given test run. First, the number of objects to be sampled is chosen uniformly from $[1, 4]$. Second, an location (x, y) is uniformly chosen from a bounding region around the given room. If this point is outside the simplicial complex defining the room, then it is resampled until it lies within. This method was chosen due to the difficulty in ensuring uniform sampling directly from a simplicial complex. Finally, a ground-truth label is chosen from that room's object likelihood model. This object likelihood model is similar to the distribution of object labels given room label learned from the Open Mind Indoor Common Sense database described in section 5.3.1. This distribution is blended with a uniform probability of finding each object type to represent that sometimes unexpected objects can be found in strange places.

The object simulator provides the correct label for recognition if the Examine action is executed within 2 meters of an object, and it provides a distribution on the object which is uniform across classes at 4 meters and becomes more peaked at the correct label between 4 and 2 meters from the target. This is an idealized model of robot performance designed to demonstrate the performance advantage of the PCM planner over an uninformed search strategy.

To establish a baseline for comparison with an uninformed version of the PCM Planner, we replaced its room adjacency and object/room affinities with uniform models. This is the PCM Planner without any context since all adjacency and object affinities are equivalent. The simulated experiments compare the time to complete an object search task using the PCM Planner with a contextual model of object-room

and room-room adjacency relationships, to the time taken by an uniformed version of the PCM Planner. For each example topology, the robot is started in an antechamber denoted as 'Foyer' and multiple runs are carried out with a different random seed to generate unique locations and labels of objects. For each random seed, the robot is tasked with retrieving one of the object classes; this experiment is repeated for each object class.

6.5.1.1 Simulation results

The simulation described in section 6.5.1 was used with a set of five rooms, shown in figure 64. Objects were sampled from a distribution similar to the one used in selecting the PCM affinity parameters between objects and rooms. Between one and three objects are placed uniformly within each room. The PCM Planner is compared with the uninformed version to locate each of the target object classes.

In each experiment, the robot was started in the Foyer, as seen in figure 64. The robot was given an object class to search for which can be found in one or more of the rooms in the simulated environment. Each object was tried three times with random variation in the specific objects sampled for each room and their positions for both the uninformed and the context-aware PCM planner. Random seeds were re-used between the uninformed and the context aware tests so that each option was tried on exactly the same arrangements of objects.

The results of this experiment can be seen in table 14. The time taken to find each object class in each run is shown. The median run time is shown in bold. Certain objects such as the Toys and Electronics can be found in the living room, which is the first room explored by both algorithms and was therefore found quickly. The advantage of using context starts to become evident with the Food class. The context-aware version of the PCM planner was able to skip examining objects in the Living Room once it had identified one object in this room. The uninformed planner had

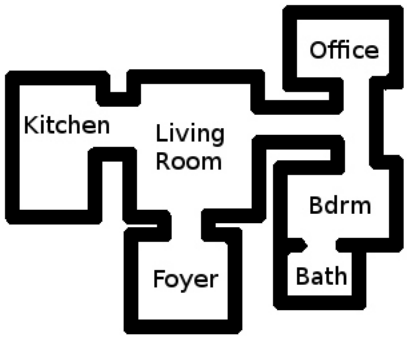


Figure 64: The layout of the world used in the simulation experiments. The robot starts in the Atrium, and is given an object class to find. The robot must discover the topological structure of the environment to leverage contextual information to improve the search procedure.

Target	PCM Planner	Uninformed	Improvement
Food	69.5, **84.2**, 85.2	92, **118.8**, 504.5	29%
Toys	42.7, **42.9**, 45	42.6, **43.2**, 46.7	0.7%
Sundries	344.7, **411.5**, 417.2	369.7, **534.3**, 565.8	23%
Electronics	42.4, **44**, 46.4	42.4, **44.1**, 46.2	0.3%
Tools	289.6, **354.2**, 650.6	429.7, **529.2**, 652.3	33%
Cooking	73.9, **111.6**, 125.8	128, **130**, 163.7	14%

Table 14: The results of the simulation experiment. Entries are the number of seconds taken to find the target object class on three runs. Median values are shown in **bold**.

to examine each object in detail before moving to the next room. Room adjacency can be seen to improve performance significantly in the hardest objects to find, the Sundries and Tools. These objects are found in the rooms which are furthest away from the entrance. In the case of searching for Sundries, the context-aware planner is able to realize that the unknown room adjacent to the Bedroom is the best place to continue to search for a Bathroom which will contain the Sundries.

6.5.2 Aware home experiment

We performed live robot experiments in the Aware Home facility at Georgia Tech [Kientz et al., 2008] using our mobile robot "Jeeves", as seen in figure 63(b). In these experiments, we placed one object on a table in each room in a conspicuous and easy to

(a) Jeeves examines an interesting object.

(b) An object from the *Toys* class is automatically segmented from the background and recognized.

Figure 65: Jeeves searching for Toys in the Aware Home at Georgia Tech using the PCM Planner algorithm.

find area, and had many un-modeled objects on other tables in the background. For example, we placed a *Food* object on the kitchen table, and we left the cups, plates, and vases on the counter for which we had not built recognition models. An image of the robot examining an object of the *Toys* class in the Living Room at the Aware Home can be seen in figure 65(a).

6.5.2.1 Results

The live robot experiments at the Aware Home succeeded in demonstrating that the modules we have developed are capable of accomplishing the object search task, and that they benefit from the context determined by the PCM. Many other experiments are currently being run at the Aware Home, therefore it is very cluttered; also, Jeeves is somewhat large and has difficulty moving through narrow doorways. For these reasons, we limited the initial Aware Home experiments to the Kitchen and the Living Room.

To illustrate one experiment sequence, the robot starts searching the Living Room until it finds an instance of the *Toys* class of objects. An image generated by the object classification/recognition module can be seen in figure 65(b), indicating that the robot has recognized a member of the *Toys* class. Once the robot found the

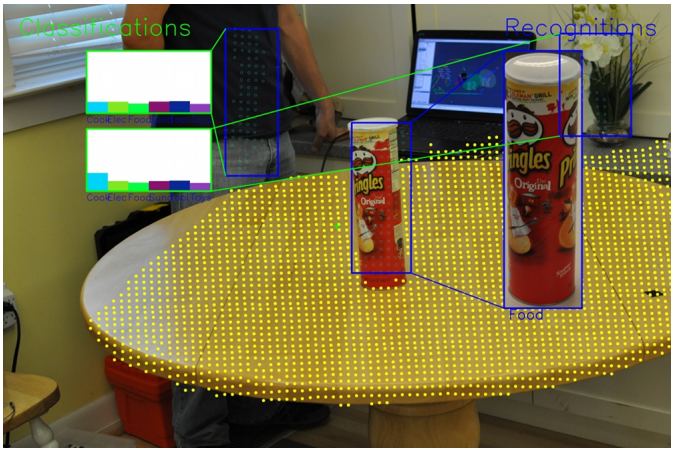


Figure 66: An object from the *Food* class is automatically segmented from the background and recognized. Additional background objects are classified.

Toys object, it was able to determine that it was most likely in the Living Room. The probabilistic model posterior belief state after this object recognition is shown in figure 67. Seeing this object has made the robot believe that it is in the *Living Room*.

The robot now has two exploration frontier hypothetical room nodes; it chooses one of them to explore. Both of these hypothetical room nodes are likely to be *Kitchen*s since the kitchen is likely to be next to the *Living Room*. The robot searches in this room and finds the target object class, *Food*, seen in figure 66. Once the object is recognized, the posterior belief state is updated, as seen in figure 68. The robot then selected this object

6.5.3 Military experiment

To demonstrate the effectiveness of the PCM planner in alternative scenarios, we adapted the application and models for an urban combat support scenario. This experiment represents about one week worth of effort to adapt software modules for use with an additional robot platform and in a new scenario. This new experiment is performed in a military training facility used to prepare soldiers for urban combat in current theaters of war. The specific location of this training facility is undisclosed

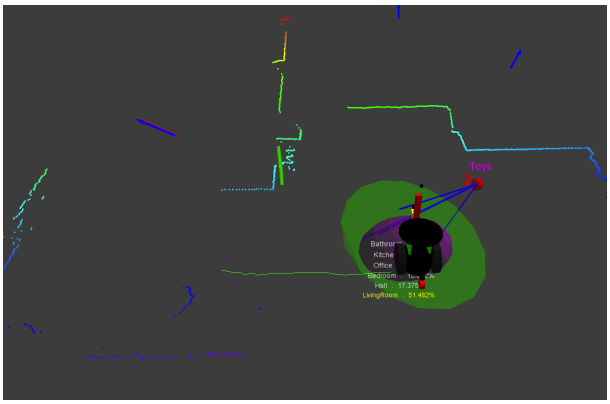


Figure 67: Jeeves has seen a *Toys* object on a table in the Living Room. The robot is searching for a *Food* object, and the PCM planner decides to search an adjacent room instead of searching the Living Room further to find this object (as Kitchens are likely to be next to Living Rooms)

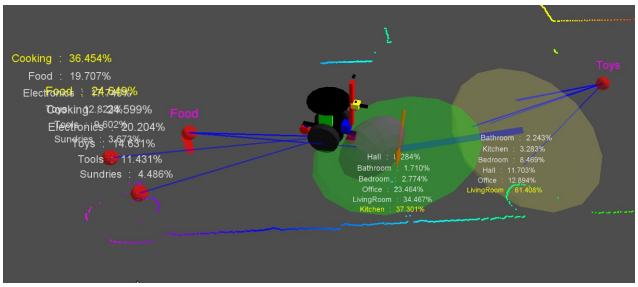


Figure 68: Jeeves has found the Food object after searching the Kitchen adjacent to the Living Room. The robot has correctly identified the Living Room and Kitchen by finding a Toys and a Food object on tables. Additionally, Jeeves has made classification measurements on two objects on a shelf behind the kitchen table where the Food object lies. Thin blues lines indicate where object measurements were made for SLAM. Text below large ellipses indicate room label posterior distributions, yellow indicates the most likely element (Living Room is 61% where Toys is found, connected to it is a Kitchen with 37%). Pink text indicated objects which have been recognized. Other text above objects indicate object posterior distributions (two are shown overlapping). The robot believes that these objects are either for Cooking or are Food due to the context of being in a Kitchen.

in this thesis for security purposes.

The US Army Research Lab (ARL) Micro Autonomous Systems and Technology (MAST) project has as its objective: *Perform enabling research and transition technology that will enhance warfighter's tactical situational awareness in urban and complex terrain by enabling the autonomous operation of a collaborative ensemble of multifunctional, mobile micro-systems.* Our research thrust in support of this mission is to develop collaborative mapping and exploration software for teams of robots to provide *situational awareness* in an urban combat scenario. This software is ultimately intended to help soldiers explore and clear suspected insurgent holdouts with minimal risk to personnel and civilians.

IRobot PackBot robots are deployed along with other small unmanned ground vehicle (SUGV) platforms in current theaters of war such as Afghanistan and Iraq. These platforms are currently tele-operated by at least one soldier to perform a variety of roles. The most important role currently undertaken by tele-operated SUGV platforms is explosive ordinance disposal (EOD). In this scenario, a robot is usually tele-operated to operate on an improvised explosive device (IED) or roadside bomb to disable its detonator mechanism with a small explosive charge or other means. Other roles for SUGV platforms in theater include reconnaissance; the robot is sent in ahead to check for hazards before committing troops into a situation. SUGV platforms are also being equipped with anti-sniper mechanisms which can triangulate the origin of enemy fire against a platoon. Finally, these robot platforms are being equipped with weapons to provide an accurate and steady firing position which may be too unsafe for a soldier to occupy. These platforms are able to virtually eliminate risk to personnel in these scenarios while performing these critical tasks.

The current effectiveness of SUGV platforms in these roles is limited by the inconvenience of deployment and management. Often, several soldiers are needed to operate a single robot; therefore, the use of robots can actually reduce the potential

effectiveness of a platoon. A more effective situation would be where one soldier is able to operate many robots. To address these problems, the Army Research Laboratory (ARL) is working to increase autonomy of SUGV platforms. One way to increase autonomy is to give soldiers high-level waypoint control; the robot uses its onboard sensors to avoid obstacles while reaching its goals. Another way to increase autonomy is through exploration algorithms which direct the robot to build a map of an unknown environment quickly. The ultimate goal is to deploy a fully autonomous SUGV which can operate in an unknown environment to accomplish mission objectives. Our goal will be to enable a SUGV platform to search an unknown environment to find a target object which is relevant to an urban combat scenario.

The scenario envisioned for these experiments is one where a soldier has introduced a PackBot into a potential insurgency holdout to search for weapons of mass destruction (WMDs). These experiments were performed at a military training facility in collaboration with my colleagues at the Army Research Laboratory (ARL), through the MAST CTA.

The iRobot PackBots used for autonomy research at ARL have been augmented with on-board computing and sensor payloads. These platforms make use of the ROS system, therefore the PCM Planner and other software components detailed in this thesis were easily adapted.

A custom sensor head was constructed to mimic the one on Jeeves. This sensor head consists of an Asus Xtion Pro Live 3D camera and a Point Grey Research *Chameleon* with a long focal length lens for foveated vision. This camera is of considerably lower resolution (1.2 MP) than the Nikon D90 used on Jeeves (12 MP); however, the object recognition algorithms are still able to perform well. An image of the PackBot can be seen in figure 69(a), and a close-up focusing on the sensor payloads can be seen in figure 69(b).

Re-training room adjacency models and object/room models was infeasible due

(a) An iRobot PackBot poised to enter a building to search for WMDs.

(b) A close-up view of the robot's sensors.

Figure 69: The iRobot PackBot used in these experiments. This machine has been augmented with an onboard computer, a Velodyne 32E LIDAR, and a pan-tilt unit with an actuated visual sensor head consisting of a foveated Point Grey Chameleon camera and an Asus Xtion Pro Live 3D camera.

to the fact that this information is inaccessible due to security concerns. Informal discussions with a small group of soldiers allowed us to update the existing models for this specific task. During their deployment in the current theaters of war, the consensus was that improvised explosive devices (IEDs) are typically found under construction in the kitchen area in insurgent residences due to the availability of water and drainage useful for processing chemical compounds. In addition, indoor bathrooms are uncommon in poorer and older communities.

Figure 70: An object built to represent the "WMD" class of objects. This model is the target object that the robot will be searching for in the military experiments. Usage of a real WMD or roadside bomb was avoided due to safety and security concerns.

Model parameters were modified to include the additional data given by the soldier interviews. First, the bathroom was removed from the room adjacency model. Finally, an additional class of objects was added: the *WMD*, or Weapon Of Mass Destruction. This object was added with a high affinity for being found in the kitchen and a slight affinity for being found in the living room. This object class has only one object instance, which is a specially constructed WMD simulation object that can be seen in figure 70.

6.5.3.1 Experimental protocol

The military scenario experiments are performed to demonstrate that the algorithms and modules described in this thesis can be extended to new platforms and operational goals. The military training facility consists of a small village with various type of buildings such as a hotel, church, hospital, embassy, and house. The house was chosen since the operation we are simulating is performing a search for WMDs (or IED components) in an urban combat scenario.

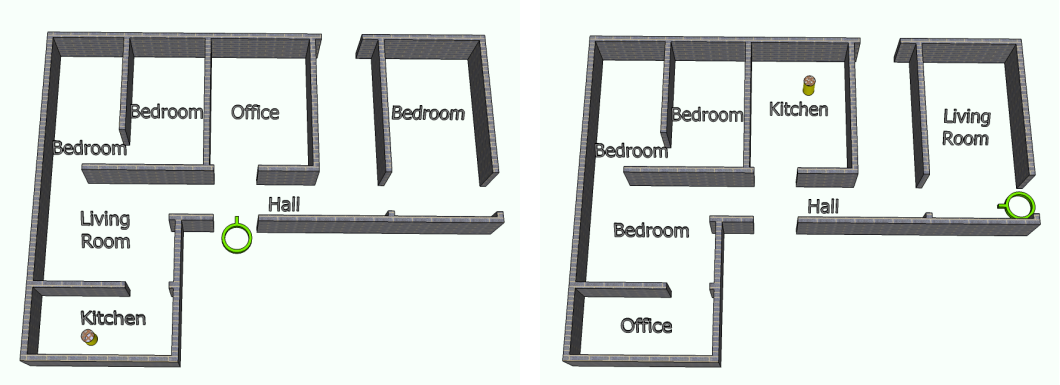
Scheduling constraints allowed for only simple proof-of-concept experiments to be performed at the military training facility. Three runs were performed from each of two starting configurations. Ground-truth room labels were chosen and relevant objects were placed in the rooms for the robot to find. The room labels were chosen differently between the two test runs to prevent the robot from finding the target object too quickly. The Office room from test1 is probably more like a Kitchen; however, it is the first room that the robot explores when it is introduced from directly across it in the hall, as seen in figure 71(a). For this run, the Office in figure 71(b) is switched with the Kitchen to put the goal object farther away. This forces the PCM Planner to leverage initial exploration results to make decisions rather than randomly stumbling upon the target object.

6.5.3.2 Results

The first starting configuration can be seen in figure 71(a). The robot is introduced through the front door into a hallway. In each run, the robot detected erroneous objects at the door jambs leading into the Office. This was due to a limitation of the object segmentation system discovered as it was extended to be used on objects on the floor. Care was taken to prevent false object detections from being made on walls. Since the robot has a limited field of view and range, small pieces of wall were segmented as objects above the floor, typically in the periphery of the robot's vision. This was mitigated by slightly contracting the convex hull of floor plane segments toward the plane centroid. Unfortunately, this did not handle door jambs when the robot could see partially around behind the jamb. In this case, the contracted convex hull still sat below the jamb and it was segmented as an object. One technique that can mitigate this shortcoming is to use concave hulls for plane representation; this is an active area of development but was not finished by the time these experiments were performed. The presence of a few false objects is of little concern since the object recognition modules correctly found no matches to the object database and these objects were discarded.

In each run in the first configuration of figure 71(a), the robot first proceeded into the office. A Netgear wireless access point was placed on the floor. This object was specifically not in the recognition database; however, it did not matter since the robot failed to notice it in any of the runs due to its small size. The robot spends some time examining other objects such as radio equipment and bundles of wire. Since it doesn't find anything which indicates what the room label is, the robot performs an exhaustive search on all the objects in this room before proceeding to the next area.

In each of the three tries, the robot enters the next useful area: the Living Room. This room is partially demolished and has large boulders on the floor, which made navigation difficult. In this room, the robot finds a Toy object which is a decoration.



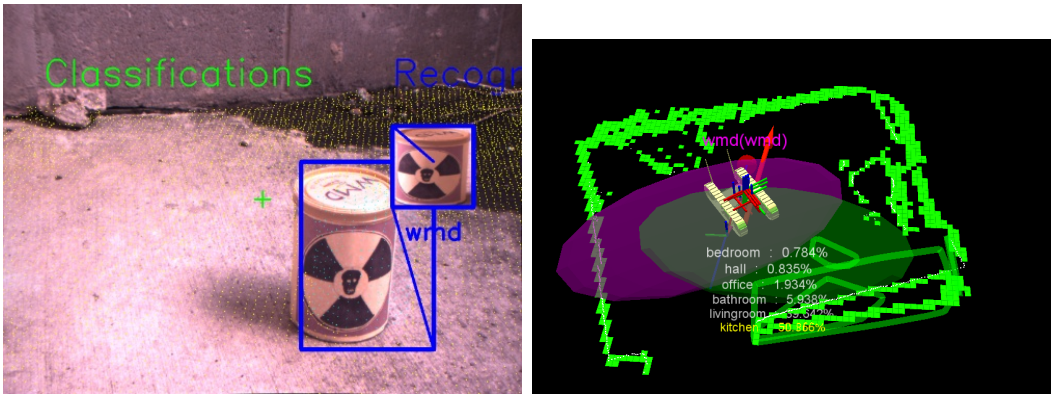
(a) The first test arrangement. The robot is introduced through the Front door, and the *WMD* is found in the room at the lower-left.

(b) The second test arrangement. The robot is introduced through the Side door, and the *WMD* is found in the room at the top-center.

Figure 71: The floor layout of the House building at the military training facility in two arrangements. The robot is drawn as a green circle with a line pointing in the direction of facing. The goal object of the *WMD* class is found in the Kitchen.

In some runs the robot also detects some medium sized rocks as potential objects but does not misclassify them. When the Toy object is recognized, the robot realizes that it is either in the Living Room or the Bedroom; therefore it proceeds to an adjacent room to further its search. The robot enters the Kitchen in each of these runs and finds the WMD in two out of the three runs. In the other run, the robot knocks over the WMD as it is entering the room and fails to recognize it. The recognition result and final state can be seen in figure 72.

The second starting configuration can be seen in figure 71(b). The robot is introduced through the side door and into the hallway next to the Living Room. The goal object is again placed in the Kitchen; however, in this run the Kitchen has been moved to a room which is more kitchen like where the Office was in the prior experiment of figure 71(a). In each run, the robot explored the living room until finding a Toy object. When this happened, the robot proceeded to the hallway and into the Kitchen to search for more objects. Once the robot had found the Toy object, the room it was in is likely to be a Bedroom or a Living Room – these rooms are unlikely to contain WMDs, so the robot chooses to explore other rooms.



(a) The WMD object is identified by the robot.

(b) The room is identified as a Kitchen at 51%. This image was enhanced for clarity.

Figure 72: The goal object of the *WMD* class is found in the Kitchen

In each of these three trials, the robot proceeded into the Kitchen. In the first trial, the robot identified the WMD successfully. In the second trial, the robot knocked over the WMD object during the search procedure and was unable to recognize it. In the final trial, the robot successfully segmented the WMD object but was unable to recognize it. This experiment trial was too late in the day and there was insufficient light to recognize the WMD object.

6.5.3.3 Analysis

These experiment runs achieved a 50% success rate after a few days of frantic preparation. This performance is clearly insufficient to think about fielding a system based upon this technology at its current state. These experiments succeeded in demonstrating that this technology is adaptable and generalizable to additional applications in mobile robotics.

Many of the run failures related to problems with removing the assumption that objects are found on tables in the environment. Without the strong table cue, it is much more difficult to find objects. Thresholds had to be set so as to filter out too many false positive objects which caused the system to miss the desired object. Also, navigation height thresholds were set too high which allowed the platform to

strike some of the objects while performing search operations. Finally, the Point Grey camera is of much lower resolution at one megapixel (compared to the DSLR at 12 megapixels) and does not have a flash or onboard light source. This camera had poor low light performance and failed to provide a sufficiently detailed image in the final test run at the end of the day.

6.6 Discussion

We have presented a technique for leveraging contextual cues in the form of room adjacency and object in room affinity in a Markov decision process based planner framework. The problem addressed in the experiments in this chapter is to find an element of an object class in an unknown indoor environment. This technique was evaluated with a simulation which simplified the object recognition and classification components and established the usefulness of the technique over an uninformed search. A live robot experiment was presented in which our robot was able to use the planner to leverage these contextual cues to find the target object class. Contextual cues were learned from training data and were applied to improve robot performance in an object search task.

Currently, all probabilistic cognitive model edge affinity and node prior parameters are trained from static offline data. We would like to augment this form of training with an online component to adapt to particular surroundings.

In many ways, the scenario chosen for this chapter is more difficult than a domestic service robot would encounter. In this chapter, the robot starts with no information about its surroundings – it must construct a new map and figure out what the room labels are each time. If the robot was able to load a previously built model, or if it were given the labels for the rooms, then the object search task could be performed much more quickly. Both of these alternatives should be considered; a robot loading a previously built model corresponds to a system which has been operating in a

home for a long time and has accumulated information about the environment, and a human labeled model represents a system which was provided with an initial tour of the home by its new user. Both of these scenarios are realistic possibilities for domestic service robots.

In the military scenario, exploration, search, and mapping of potential insurgent holdouts would probably benefit from additional tactical knowledge such as aerial or satellite imagery. Additional domain knowledge beyond what was received from informal soldier interviews could be used to improve the PCM representation for this scenario. Finally, a system for sliding autonomy where a soldier and a robot work together could achieve much of the benefit from an autonomous system with limited distraction for a soldier.

VII

DISCUSSION, CONCLUSION, AND FUTURE WORK

7.1 Introduction

A global representation for the location and identity of objects can be used by a robot to perform a variety of service tasks. Many of the chores which domestic service robots will be expected to perform involve objects; therefore, a representation which tracks the location of these objects is useful. People organize the goods in their homes according to the purpose for which they are used; many rooms are specialized for performing certain tasks and will usually contain the objects associated with these tasks. This organizational paradigm provides a valuable contextual cue which can be leveraged by a mobile robot to perform its duties.

Context is a concept which is loosely defined as the circumstances which surround a particular situation. Robots can use context by understanding aspects of these associated circumstances; to provide information about an unknown or unseen situation. This thesis described a *framework* for reasoning or making inference based upon contextual cues surrounding objects in structured environments.

By mapping the location of objects and rooms, contextual relationships between objects and places can be identified. Recognizing unique objects can also provide strong cues for *data association*, if the object being recognized is static or relatively permanent. Mobile robot mapping navigation can use this data association cue to recover from the *kidnapped robot problem*, and object recognition can be informed by the contextual relationships between other entities uncovered through mapping. Using objects and context allows mobile robot mapping and object recognition to leverage one another and improve performance.

We have presented a methodology for incorporating various types of sensor information in a model for reasoning about context in mapping and object recognition. This methodology, dubbed the *probabilistic cognitive model*, represents the belief state for a mobile robot over object and room labels as it explores an unfamiliar environment. This model is trained from examples of floor plans and crowd-sourced online databases of common sense[1]. The model is based upon a probabilistic graphical model where nodes are objects and rooms, and edges represent topological associations between these entities. Underlying this model is a system for metric and topological mapping called the *OmniMapper*. A planner which uses this model to represent its belief state and hypothesize the results of its actions was developed to control mobile robots. This planner is used to field two robot applications, a domestic service valet who searches for an object in your house, and a military robot which clears an insurgent hideout while looking for "WMDs"(or improvised explosive devices).

In the domestic service valet and military robot applications described in this thesis, the robots are required to explore an unknown indoor environment while searching for a target object (or object class). In the military counter-insurgency "WMD" search application, this is a fairly realistic scenario; however, a real implementation might be provided with more information such as aerial imagery or covert ground surveillance. The domestic service valet application will only have to explore an unknown environment when it is first switched on; after that its prior experiences and saved maps can be used to shortcut the exploration phase of the object search task.

Due to the fact that these robot applications start in unknown environments, they start their experimental test runs by searching the room they start in. If an object is found and recognized in this room, and it is not the target object the robot is searching for, then the inference process allows the robot to choose whether it should continue searching this room, or it should move on to an adjacent room. While searching for

[1]Open Mind Indoor Common Sense Database

an object like *vitamins*[2], if the robot identifies a television remote control[3], then it will immediately stop searching in this room and move to an adjacent one. Moreover, the robot will also prefer to look in rooms further away from this one, since Living Rooms or Offices are unlikely to be next to Bathrooms, and these would be the most likely room labels. This is the mechanism through which the object search task was achieved more quickly with an context-aware model as compared to an uninformed model.

7.2 Thesis summary

Chapter 2 presented some key developments we have made for life-long mobile robot SLAM. The first two studies in this chapter address two of the main problems associated with long-term operation of a mapping system. The first problem is that of managing complexity: any mapping algorithm will use more computational resources as more information is gathered from the environment. The first study in this chapter presented a technique for managing this complexity by *approximating* the final map by removing information. The information removed from the map was chosen via a *normalized graph cuts* algorithm which minimized the difference between an approximate map and the full map. The second problem is how to handle dynamic landmarks. When a robot operates for a long period of time, it will encounter various aspects of its environment which change. The second study in this chapter presented an *expectation-maximization* algorithm for determining which objects in the environment were static and which were *moveable*. This study allows for the classic *static world assumption* to be relaxed; the *static world assumption* is the assumption that the operating environment around the robot does not change except by the robot's actions.

[2]This object is in the Sundries class, typically found in the Bathroom, and sometimes in the Kitchen

[3]This object would be in the Electronics class, typically found in the Living Room or the Office

There are many type of environments in which a robot might need to build a map. We developed an extensible framework and mapping methodology, called *OmniMapper*, which uses a system of plugins, generic programming, and the *M-space* formulation to build maps in a variety of environments with different combinations of sensors and landmarks. The *OmniMapper* builds a metric map of the robot's environment while determining the robot's location, identifying open areas where the robot can navigate, and answering queries about landmarks.

In the final study presented in chapter 2, we looked at how feature-based mapping with the *OmniMapper* enables robots to use lower-performance sensors without impacting mapping and localization performance. This study is important because the scientific sensors used in research today are too expensive to place on consumer robot equipment.

Since a single robot appliance is unlikely to work alone in the home-of-the-future, we described how robots can work together to perform mapping tasks in chapter 3. The *OmniMapper* library was extended to work across teams of robots to perform multi-robot mapping. We presented several studies which demonstrated different algorithms for sharing information among a team of mobile robots, as well as the strategies they use for collaboration. Since a variety of differently equipped robots can accomplish a wider variety of tasks, we presented a study demonstrating techniques for fusing information across heterogenous teams of mobile robots for map building.

In unknown environments under autonomous operation, robots must build a map while looking at that map to find places to explore. In the multi-robot exploration and mapping problem, members of a team of robots work together to autonomously build a map of an unknown environment. The final study in chapter 3 compares some strategies which can be employed by a team of mobile robots to coordinate their exploration efforts.

Even the best robot mapping and navigation system might become lost if it encounters a pathological environment, suffers from sensor or actuator failure, or is mysteriously transported to a foreign location. We presented a technique called *semantic data association* in chapter 4 which uses high-level reasoning about *Object recognition* and more meaningful landmarks to allow the robot to *re-localize* itself when it becomes lost. The first study used a learned classifier to identify a particularly useful visual landmark: door signs. Finally, this chapter concluded with a description of the techniques which were developed for identifying, recognizing, and classifying objects for use in mapping.

A representation for context between objects and places was presented in Chapter 5. This representation, dubbed the *probabilistic cognitive model*(PCM), uses a probabilistic graphical model to couple object and place recognition and classification. The PCM is built from a *topological* map of objects and places which is linked to a *metric* map built by the *OmniMapper*. The topological map representation contains *places* with their adjacency relationships to other *places*, and *objects* within those *places*.

Finally, we presented a methodology for using the PCM representation with a planner in Chapter 6. This planning methodology used the PCM to represent the *state-space* in a Markov decision process, with which the robot is able to evaluate what changes will result from its actions. This planner was first used in a simulated experiment which demonstrates that the use of context in the PCM is more effective than an uninformed strategy. Two real-world live robot applications were also presented. The first application was a domestic service robot valet, called "Jeeves", who uses the PCM planner to explore an unknown home to find an object. The second application was a military robot which searches for WMDs in an urban counter-insurgency scenario.

7.3 Main results

The specific lessons learned during the development and testing of the methodologies detailed in this thesis include:

- Probabilistic representations such as the *probabilistic cognitive model* are ideally suited to representing the effects of contextual reasoning in the real world. Mobile robots operate in an uncertain environment where little can be taken for granted. By maintaining probability distributions over entities, a mobile robot can cope with this uncertainty.

- The fact that we are able to use multiple types of sensor modalities allowed us to deploy robots running *OmniMapper* in a variety of challenging environments. *OmniMapper*'s plugin infrastructure is adaptable; it enabled us to use the many platforms and sensors in the experiments presented in this thesis. The environments which can be mapped by *OmniMapper* include those with structure which can be recognized as features, as well as those absent any regular structure. In this way, *OmniMapper* can be used to map most environments, with the proper sensory feedback.

- Experimental evaluation of the *probabilistic cognitive model*(PCM) planner has taught us that it is a useful technique for leveraging context to guide a mobile robot in an uncertain environment. A robot system using this planner can explore a home and understand the purpose of each room; this significantly reduces the effort and time needed to locate an object of interest.

- The experimental evaluation carried out in this thesis focused on searching for objects in an unknown environment. In a domestic service role, a robot will usually have prior knowledge about its environment. This prior knowledge can be used by the *probabilistic cognitive model* to further accelerate the planner's search to focus on areas more likely to contain the object of interest.

7.4 Future work

The types of context which were explored in the current structure of the *probabilistic cognitive model* are limited to an object and place *within* relationship and a place *adjacency* relationship. The use of conditional random fields for the computation of the PCM state allows us to use additional types of contextual relationships. A simple improvement would be to use the place classifier from Mozos et al. [2005] to help classify places without relying on object recognition. Also, contextual relationships between objects could be used such as pairs of objects which are related might be close together. For example, the computer mouse is *near* the keyboard. Once the keyboard is recognized, the objects nearby should be searched closely to find the mouse. More complex predicate relationships could also be added which rely on spatial reasoning, such as *above*, *below*, *on*, *to the right of*, and many others. Since the *OmniMapper* determines these relationships, these between-object relationships could be added where the appropriate spatial relationship exists.

With the exception of object recognition modules and loading and saving maps, the current system does not adapt its internal PCM parameters to the robot's specific surroundings. The model parameters are trained from sources which represent the typical home, and typical objects. A very strange environment, such as in figure 73, would require online adaptation of the PCM model parameters to capture its atypical arrangement. Also, if the robot's master does atypical or unsanitary things like eating in the bathroom, the robot would need to adapt to this in order to know to look for dirty dishes there. Online adaptation was not developed for this thesis, but it could be incorporated by the robot collecting ground truth labels for rooms and objects, and then performing a stochastic gradient descent update on its model parameters with this new training example. Since a domestic service robot should be personalized for a specific user's home, the generic model parameters should gradually be replaced with better values which represent its user's house.

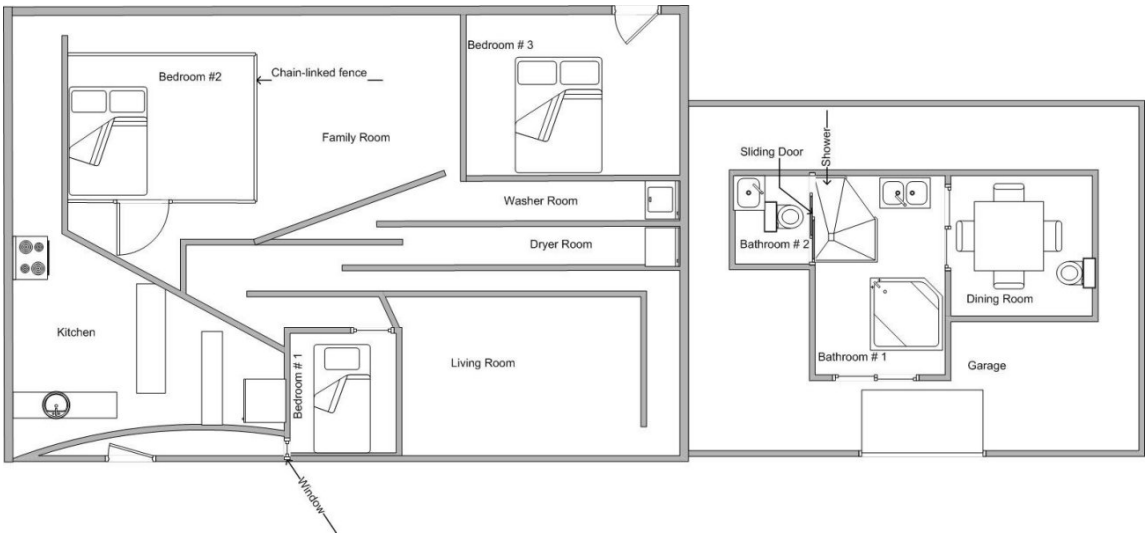


Figure 73: This floor-plan would require online adaptation by the PCM. Image courtesy of user named *kftwin*, found on www.reddit.com.

The PCM planner is currently used to control a robot to search for an object and fetch it. The planner is focused on tasks which have a degree of *uncertainty* and might require exploration to complete. There are other basic tasks like this that could be added to the PCM Planner such as navigating to a place, like *Go to the kitchen*. In addition, the planner could perform composite tasks like *Bring me a beverage from the kitchen*. The PCM Planner could also be used with a high-level planner to accomplish more domestic tasks like cooking meals, as in Beetz et al. [2011].

Finally, the robots which were used in these applications relied on several expensive sensors which would not be available on a consumer product. The *Kinect* sensor is a cheap alternative to a laser-based ranging sensor, and is already a key component of the hardware in both of the real-world robot applications described in this thesis. It would be interesting to see if the laser-based ranging sensors could be removed and the Kinect sensors relied upon solely for mapping, navigation, obstacle avoidance, and safety. In a covert military scenario, passive sensors are preferred since both laser ranging sensors and Kinect cameras emit energy into the environment which could easily be detected by an enemy. For this reason, it is also interesting to consider how

passive sensors like monocular and stereo cameras could be used to substitute for these active sensors.

VIII

APPENDIX

8.1 SIFT and SURF comparison on object recognition and classification

Feature recognition requires two components, an interest point detector to identify regions in the image which are distinct, and a feature description which can be computed from the neighborhood of the interest point and compared with other descriptors to determine correspondence. This paper will extend prior comparative analysis to include the new techniques with a comparison on the classical computer vision tasks of object recognition and object classification.

8.1.1 Feature descriptor comparison

Mikolajczyk and Schmid [2003] compares combinations of interest point detectors and feature descriptors for object recognition performance. The types of interest point detectors covered include the Harris corner detector [Harris and Stephens, 1988], two detectors developed by Mikolajczyk *et al.* as extensions to the Harris detector called the Harris-Laplace [Mikolajczyk and Schmid, 2001] and the Harris-Affine [Mikolajczyk and Schmid, 2002], as well as the Difference of Gaussians interest point detector which the SIFT [Lowe, 2004a] to find *scale-space extrema* [Lowe, 1999] interest points.

Mikolajczyk and Schmid [2003]compare current feature descriptors on real image pairs with rotation, scale changes, affine changes (moving the camera) and illumination changes. All of these variances were prepared by altering the camera and lighting in the scene, not by computing transformations of digital images. The feature descriptors compared in Mikolajczyk and Schmid [2003] include the SIFT descriptor

[Lowe, 2004a, 1999], steerable filters [Freeman and Adelson, 1991], differential invariants [Koenderink and van Doorn, 1987], complex filters [Schaffalitzky and Zisserman, 2002], and moment invariants [Gool et al., 1996]. Mikolajczyk *et. al.* determine that the SIFT descriptor has the highest detection rate for a given false positive rate. The SIFT descriptor will be considered along with a new descriptor called *SURF*[1] [Bay et al., 2006].

8.1.1.1 Scale Invariant Feature Transform (SIFT)

The SIFT feature descriptor [Lowe, 2004a] was developed by Lowe to capture the saliency of a region around an interest point and recognize it under scale, rotation, illumination, and some affine transformations. The interest points in Lowe's paper are local extrema in *scale space.* That is, each interest point is a local maximum or minimum of a Difference of Gaussians (DoG) filtered image. The image to be analyzed is first convolved with a bank of Gaussian filters at successively higher σ values. Adjacent images are subtracted to compute a set of DoG images, and points which are maximum or minimum value among their 8 neighbors within one DoG image and within the 9 points in the next scale and 9 points in the prior scale are identified as extrema in *scale space.* SIFT achieves scale invariance by detecting interest points at the scale where they are local extrema; in another scale this same scale will likely be the local extrema again. The principal gradient in the neighborhood of this feature is analyzed and the neighborhood is rotated to move this gradient into a modeled location so that the region is now rotation invariant.

The SIFT feature descriptor is computed from the neighborhood of this scale and rotation invariant interest point as follows. The neighborhood refers to the interest point local region which has been transformed to be rotation and scale invariant. The gradient magnitude and direction is determined and averaged within 16x16 square

[1]This analysis was performed in 2006, right after the SURF descriptor was introduced

regions around the interest point. These 256 directions and magnitudes have their magnitudes weighted by a radial basis function (to reduce the effect of more distant gradients from the interest point). The weighted magnitudes and directions are combined into a 4x4 histogram with 8 directions. This gives a 4x4x8 = 128 dimensional feature vector. Illumination invariance is achieved by normalizing this vector to unit length.

8.1.1.2 Speeded Up Robust Features (SURF)

The SURF descriptor was described in Bay et al. [2006] as a faster and more robust feature descriptor than SIFT. This descriptor incorporates a new interest point detector and a new feature descriptor. According to this paper, the DoG interest point detector is in fact approximating the Laplacian of the Gaussian, which is the trace of the Hessian of the image, the matrix of mixed partial derivatives of the gradient. The Laplacian of the Gaussian filter, as it is the sum of the unmixed partial derivatives, responds strongly along edges as well as regions of maximum curvature – this causes poor localization of interest points and requires an additional computation to prune out interest point candidates which lie along a ridge. In contrast, the SURF detector employs an approximation to the Hessian of the Gaussian interest point detector by computing the determinant of the convolutions of the mixed second derivatives of the Gaussian with the intensity image. The convolutions by the second derivatives of the Gaussian are approximated by constant box filters which are computed very quickly with the integral image representation. The convolution by a box filter approximation to a second derivative Gaussian goes from hundreds of multiplications and additions to only four additive operations: simply take the bottom-right corner value in the integral image and subtract the top-right corner, subtract the bottom-left corner, then add back in the top-left corner. The determinant of this Hessian built from the four combinations of second order derivatives of Gaussian responses is the interest point

value. If this value is the maximum or minimum of its 3x3x3 neighborhood in scale space, then it is accepted as an interest point.

The principal gradient is established in the neighborhood of the interest point and a local coordinate system is placed parallel to it to achieve orientation invariance. The descriptor is a 64 element vector computed from the approximate Haar wavelet responses in the neighborhood of the interest point. The neighborhood of the interest point is divided up into 16 equal square portions which are sampled with a 5x5 regular points. The Haar wavelet responses are determined with each of 4 wavelets (horizontal, vertical, absolute value of horizontal and absolute value of vertical) on these sample points and this is normalized and then stored in the 64 dimensional vector.

8.1.2 Object Recognition

There are currently two major research directions in object recognition: Model-based recognition, and Appearance-based (or Feature based) recognition [Forsyth and Ponce, 2003, Trucco and Verri, 1998]. Model-based object recognition builds a database of 3D structural components with which objects can be described. Feature or appearance based object recognition relies on the observation of salient features from images. These features are then extracted from the test image and then compared with the database to generate putative matches. Usually some geometric consistency technique is applied to filter out structurally inconsistent feature matches. Appearance-based techniques can also employ the entire image as a feature through some dimensionality reduction techniques. The principal components of this *eigenspace* can then be used to test scene images to see if the object appears in the scene.

8.1.2.1 Model-based object recognition

Model-based techniques for object recognition rely on matching a 3D model of an object into a scene and enumerating potential matches. In Biederman [1987], the object to be recognized is broken down into instantiations of 36 classes of "Geons" which are conical sections under various distortions. These structures can be recognized from the image through five factors of appearance: collinearity, cotermination, parallelism, symmetry and curvature. Classification is performed on these component geons in a way which is robust to occlusion and viewpoint. Model based techniques consider the 3D structure of the object which allows them to be robust to self-occlusion.

8.1.2.2 Feature or appearance based object recognition

Feature-based techniques for object recognition extract some salient features from the object to be modeled along with their spatial organization [Trucco and Verri, 1998, Forsyth and Ponce, 2003]. Candidate scene images have the same features extracted from them and are compared to the model features some metric. Often a "generalized Hough transform" [Leibe and Schiele, 2003, Ballard, 1981] is used to validate that all of the features matched work together to represent the object and are not spurious false positives. Feature or appearance based object recognition will not necessarily try to reason about the 3D structure of the object being modeled; they instead attempt to remain robust to 3D pose differences through sophisticated feature recognition.

The SIFT [Lowe, 2004a, 1999] descriptor is useful for object recognition. Feature matches between a model and scene image are validated through the imposition of additional geometric constraints. In Lowe [2004b], the authors use a Hough space representation to vote for poses of the candidate object match. This filters out spurious feature matches which are not consistent with any object at a specific pose. This can be accomplished for planar objects with as few as 3 matched points. If there are 7 matched points, then a full affine transformation can be computed by solving for

the fundamental matrix [Hartley and Zisserman, 2003]. The SURF technique [Bay et al., 2006] does not specify a geometric validation step; however, this is a simple addition.

The features to be extracted from the model and scene images should permit differentiation between objects. To support *object categorization*, Fidler et al. [2006], Fidler and Leonardis [2007] *learn* what neighborhoods of simple features are present in training images in an unsupervised manner. This forms a hierarchy of parts which can be used to describe objects for recognition. At the lowest layer, these features are simple oriented Gabor filter responses. Each subsequent layer is built from combinations of the previous layer. Unsupervised learning of part combinations ensures that only the most common parts will be considered and all parts will be shared among different classes of objects. Object class differentiation is accomplished by the supervised learning step where all the parts appearing in one object are associated to a class label. Recognition is accomplished by matching parts and spatial orientations to the trained supervised object models. The most accurate match delivers a class label to the test object.

Leibe and Schiele [2003] learns a model of small image templates and their spatial configuration to support object classification. These spatial templates form a codebook of parts to look for in images to determine object category. To form this codebook, first Harris [Harris and Stephens, 1988] corners are detected and a region around each corner is extracted. This region is compared with a correlation score to other samples in the codebook and it is agglomerated with other good matches. In this way, a set of parts is grouped which can support the categorization task. The detected scene parts then vote in generalized Hough space to find the object centroid based on where they had been observed during training. Leibe used this object classification to assist with object segmentation – separation of foreground objects from the background.

Opelt et al. [2006] detect the boundary of the object for consideration by segmentation from the background. Object category for many classes of objects is more dependent on the boundary than the interior features. For example, coffee cups might have any logo on them which would not generalize across the class – the general feature is only the boundary. Their technique is not hierarchical in the composition of boundary fragments from smaller fragments; instead it relies on the boosting framework to select relevant combinations of simple boundary fragments which classify the training set well. These compositions are now weak classifiers for the object of interest – through boosting a large set of these weak classifiers a strong classifier is derived. This paper presents test accuracy equal to Leibe's classifier [Leibe and Schiele, 2003] which is said to be formerly the best; however, this is only on a single class detection.

Sudderth et al. [2005] builds an hierarchical model of how SIFT features can be shared among different objects. This model uses Markov Chain Monte Carlo sampling over a probabilistic Bayesian network to find assignments of parts to objects which are consistent with training examples. This hierarchy represents scenes at the top level, objects, then parts and features at the bottom. Features are SIFT features which have been clustered with K-means to deliver 32 typical and sharable features across all trained object classes.

Ettinger [1987] described an algorithm for recognizing objects using a boundary model composed of hierarchical boundary components. Boundary components are extracted as features based on several curvature discontinuity classes: inflection, corner, end (two turns in the same direction nearby), crank (opposite turns nearby) and bump (right-left-right or left-right-left). Parts are composed of these features which are common across some subset of training examples of this class. Parts may also contain only one feature if it does not appear significantly often in the vicinity of another feature, but is still stable across some subset of training examples of this class. The use of curvature discontinuities as the definition of parts allows for scale

invariance due to the fact that curvature is an intrinsic quantity. This is accomplished without any multi-scale training; most of the other techniques donot necessarily share parts with other scales and essentially train non-overlapping classifiers.

Krempp et al. [2002] describes a technique for learning a set of object classifiers in a mechanism similar to Fidler [Fidler et al., 2006, Fidler and Leonardis, 2007]. The parts in this paper are binary features made from flexible configurations of edges. New parts are added to the classifier sequentially; as new classes are added, if the existing parts do not cover the new class, then additional parts are learned in an unsupervised manner. The recognition task tested by the authors of this paper is differentiating Greek letters under scale and orientation changes.

The Hyper-features technique of Agarwal and Triggs [Agarwal and Triggs, 2006] is to combine simple features into a histogram and incorporate histograms into a hierarchy of descriptor vectors. In this way, hyper-features extract spatial co-occurrence information of simple features while allowing for perturbations in detection and precise location. Hyper-feature stacks are naturally robust to occlusions due to their *loose agglomerative nature*.

The term *eigenfaces* was proposed by Turk [Turk and Pentland, 1991] to describe a PCA based technique for recognizing human faces in a scene. To generate the face classifier, the authors took a large number of images of human faces. These images were warped so that the eyes and mouths appeared in canonical image locations. By representing each image as a vector, eigenvectors are computed from principal component analysis(PCA). The top N principal components are extracted and can now be used for recognition. The projection of a test image onto these principal eigenvectors now will measure the test image along this salient dimension of facial change. This is a technique for recognizing a specific person by comparing the lower dimensional dot products along the eigenvectors to a database, but it is probably not the best technique for deciding if a face (or any object) is in the scene. Since this

technique relies heavily on carefully registered images, looking for a face in a scene image does not even really make sense; all faces must be carefully aligned before being projected onto the basis vectors. This algorithm fails under lighting changes and any deviation in the pose of the object to be recognized. Appearance-based techniques often do not reason about the underlying geometry and therefore have difficulty with non-rectified images as well as differences in illumination.

An alternative technique was proposed by Viola and Jones [Viola and Jones, 2004] which uses boosting on a set of weak classifiers based on Haar wavelets to generate a strong classifier. Just as with eigenfaces, their application was face detection. The use of boosting helps their classifier incorporate many separate types of faces and does not require careful alignment; this is because each weak classifier is independent of its peers where with eigenfaces, all of the training examples were averaged together. This technique is useful for finding objects without rotation invariance, as in face detection in camera imagery. In principal this could be trained with significantly different poses but it would require a very large set of training data to achieve the same scope of invariance in feature based methods.

8.1.3 Procedure

The SIFT and SURF feature descriptors are compared in terms of both object recognition and object classification. These feature descriptors are evaluated on the ETH-80 data set, shown in figure 74.Two potential applications of feature based object description are explored : object categorization and object recognition. The experiments for the object categorization and recognition tasks each feature a separate subset of the ETH-80 data set. This data set features views of each object at regular intervals of the upper viewing hemisphere, so it is ideal for testing object recognition under different poses. The background is simple and can easily be segmented and removed from the feature detection process. This data set also supports the object

classification application because it features 10 instances of each of 8 object classes. If the best match to a test object while withholding the rest of the instances of this object is still within this object's class, then it is said to be classified correctly. The recognition task simply allows for other images from this instance to be included during the matching process, though the specific test image is withheld from the training set. The recognition task uses all the remaining images as test images, while the classification task also excludes the images of this specific instance, requiring a successful match to come from another instance of that object class.

The object recognition task test procedure is as follows. Each image in the test set is compared to each of the other images within this same object type (including other images from this specific instance of the object type), as well as with images from the other object types. Each group of objects is balanced to contain the same number of images. To reduce the amount of image comparisons, a subset of the ETH-80 data set was chosen which features only the views of the object at the 66 degree viewing declination. All azimuthal viewing directions are included at this declination. Since there were 8 images for each object, and 10 objects per class, and 8 classes, there were a total of 640 images. Comparing all of the images would require about 400,000 runs of the detectors per experiment, and since the SIFT detector takes about 1 second per comparison, this would require about 5 days of computation per experiment. To reduce this amount of computation without significantly impacting the results, I have chosen to split the total recognition task into many smaller experiments on specific instances of each object. For each sub-experiment a training instance and a test instance is chosen across all objects. Using a test instance of 1 and a training instance of 1 results in a recognition experiment, while a test instance of 1 and a training instance of 2 results in a classification experiment since the objects being compared do not come from the same instance. If the best match is still within the object class, then the classification is successful. There are therefore 10 times

Figure 74: ETH-80 data set, all objects at declination 35, azimuth 45. From left to right: tomato, pear, cup, cow, car, apple, dog, horse

as many classification experiments as recognition experiments and all of the data is accumulated into confusion matrices for analysis.

The object classification task test procedure is very similar to the recognition procedure with the added constraint that the images of the specific instance of the test object are excluded from the training set. This prohibits the test object from matching on any specific feature unique to this instance and therefore only admits matches based on more generalizable features. Note that since this is an 8 class problem the test examples are balanced across each class; a random guessing approach will only achieve 12.5% recognition accuracy instead of the typical 50% recognition accuracy from a two-class problem.

8.1.4 Results

The SIFT and SURF detectors are compared below on recognition and classification of the ETH-80 data set.

8.1.4.1 SIFT detector

The SIFT detector finds many good quality interest points which can be matched to identify the same image under affine transformations, as can be seen in figure 75(a). Unfortunately, most of the features will not be matched when the image undergoes a non-affine transformation. In many cases; however, the object can still be identified by the robust features which remain invariant under this object transformation, as can be seen in figure 75(b).

8.1.4.2 Recognition

On the recognition task, the vanilla SIFT implementation achieved a 59% accuracy on the recognition task as shown in table 16. Several adjustments were made to attempt to improve the performance on this task, the first of which was to mask out the background and ignore any SIFT features which are detected off of the object. This alteration actually dropped the recognition accuracy to 51% as can be seen in table 17. Many of the interest points detected on the objects, especially the Fruit superclass, arose from the external edges located on the background side. By removing these features, some of the objects are left with fewer total features and a spurious match becomes more likely.

To confine the matches to ones which are spatially consistent with the training image, a homography was computed from the putative matches robustly using RANSAC. Unfortunately, this further diminished the recognition accuracy to only 29% as can be seen in table 18. As can be seen from this confusion matrix, the Apple object was the best match for a large proportion of the total objects examined. This is due to the fact that the Apple objects are compared first (in alphabetical order)

and superior matches are only taken if they match with more parts. Something which matches with zero parts would therefore be said to match with the Apple when in fact the software could instead have easily recognized that it was unable to determine a match. In these cases the system is forced to make a guess as to what object matches best even though it could more robustly admit that it has no idea of which object matches to this example. Also, using a homography in this way is only valid if the object being modeled is planar – all the features which are inconsistent with this planarity assumption will be eliminated by the RANSAC homography leaving us with fewer total matches. Clearly many of the objects in this dataset such as the Tomato are, in fact, not planar.

8.1.4.3 Classification

For the classification task, the vanilla SIFT implementation achieves a 29% successful classification rate as can be seen in table 19. An example set of matches for the classification task can be seen in figure 75(c). This result seemed quite poor, so the masking technique was employed to eliminate background features, assuming perfect foreground segmentation. Rerunning the classification task with the foreground mask to eliminate background features further reduced the classification rate to 25% as can be seen in table 20. In post-analysis this result is reasonable because many of the features have interest points which are located at the object boundary or in a concave region bounded by the object. This feature descriptor computed at this point still contains a great deal of information about the object to be classified – also this information is boundary details instead of internal structure, which will be argued to generalize better to entire object classes. Masking out the background features eliminate about half of the boundary features, which explains why the classification accuracy has been reduced by this operation.

The application of a robust homography to the classification task further reduced

the classification accuracy to 18% as can be seen in table 21. This also makes use of the mask to remove exterior boundary features, and also applies a homography to filter-out inconsistent feature matches. Unfortunately, this homography filters for feature matches which are not consistent with the projective transformation of a planar region, which again removes features which aren't coplanar. Most objects are now matched to the initial classes (Apple, Car, etc), which means that they are only matching with zero or one features. This technique is very good for removing spurious matches, but the conditions for being spurious are tied to the object geometry which isn't planar on any of the objects in the data set. In the recognition task with affine transformations, this RANSAC homography would generate very good results. However, the parts recognized by the feature matching technique do not always correspond to a spatially consistent organization; sometimes similar looking parts are recognized in a different order than in the training image. These matches are then lost by the RANSAC homography.

8.1.4.4 SURF detector

The SURF detector [Bay et al., 2006] generates a large number of very distinct features as can be seen in the self recognition image in figure 76(a). This vanilla detector and matcher suffered from the problem of many features matching to one feature as in figure 77(b). This caused most objects to match to the Pear object since it has boundary features at many curvatures – structures which matched in part to each of the fruit objects. By allowing multiple matches, the detector repeatedly picked the feature with the curvature for that fruit, effectively making it seem like that feature appeared many times also in the Pear. This really happened with all of the fruit, rendering them indistinguishable; however, the Pear has more features and textures than the other fruits. The author prevented more than one match at a time to happen with any feature; this made the detector much more accurate at recognition

and classification.

8.1.4.5 Recognition

SURF detector recognition example at figure 76(b). The SURF detector (with the bijection modification) achieved 81% accuracy on the recognition task as can be seen in table 22. Most of the mistakes come from the objects in the Fruit category getting misclassified within this superclass. The stem on the Pear and Apple is erroneously matched several times, and the shape of the Apple gets confused with the Tomato. The Pear also gets selected as the best match several times, even with some of the Animals, likely due to its variety of curvature at the boundary and internal texture.

For completeness of analysis in comparison with the SIFT detector, this experiment was also run with a mask to block out features which are located in the background. It was observed that many times the features are actually located in the background when they stand for structures at the boundary of the object, so it was anticipated that the recognition accuracy would drop. In fact the accuracy did drop to 70% as can be seen in table 24 when the mask was added to the SURF matcher. The RANSAC homography was not compared on the SURF detector due to time constraints; however, it is very likely that this would only further reduce the effectiveness of this detector on the recognition task.

8.1.4.6 Classification

The SURF detector also performed well on the Classification task at 39% as can be seen in table 23. The Pear object absorbs a lot of mistaken matches due to its many boundary curvatures and internal texture. Also the Apple object is often mistakenly matched to other objects due to low numbers of total matched features – a condition which can easily be detected and can result in a null match instead of a mistaken match. The mask technique was also attempted with the SURF detector, which resulted in a reduction of recognition accuracy to 31% as can be seen in table

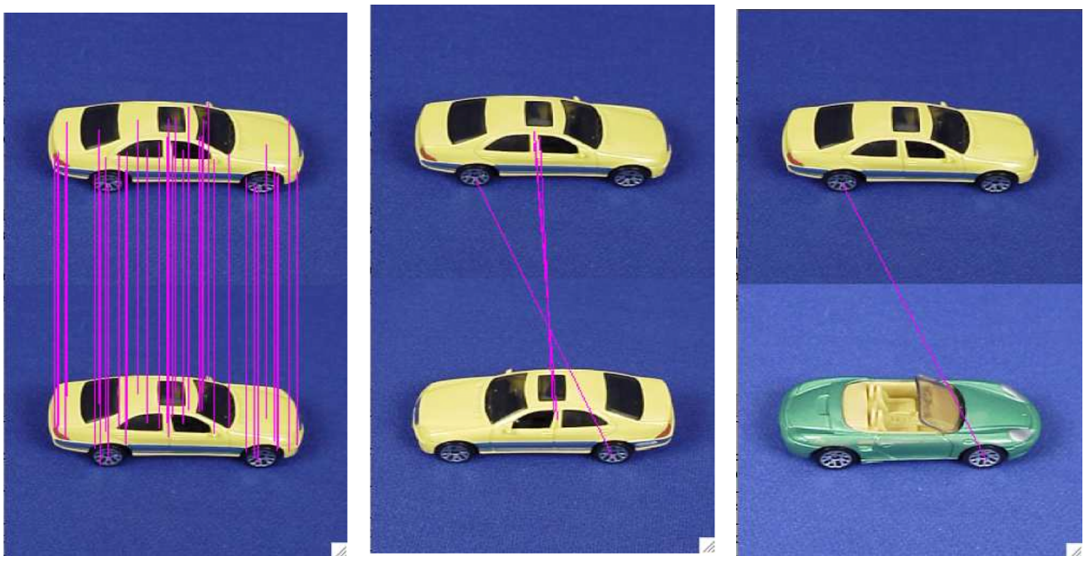
	SIFT	SIFT-M	SIFT-R	SURF	SURF-M
Rec.	0.590	0.509	0.292	0.806	0.702
Clas.	0.291	0.248	0.181	0.391	0.307

Table 15: Results for SIFT and SURF detectors with various options on the Recognition and Classification tasks on the ETH-80 data set. The "-M" indicates with Masking, and "-R" indicates with RANSAC homography.

25. The RANSAC homography was not included due to time constraints, though it would likely have only further reduced the accuracy.

8.1.4.7 Comparison

The SURF detector has outperformed the SIFT detector on both the object recognition and object classification tasks. Table 15 shows a comparison of the overall results of all tests. Many man-made objects show repeated sub-components such as the wheels of the car. Feature based methods will not necessarily match the spatially consistent set of features from different views of the same object – for example in figure 76(b) the wheel matches switch sides while other matches stay on the same side. To generate a result consistent with a planar homography, the disagreeing feature matches must be removed which may result in fewer matches and reduce accuracy. The fact that such different views would be seen of the test objects means that enforcing this spatial constraint will not always boost the accuracy and often removes very good matches of individual parts which do a good job of classifying an object. Full reasoning about the geometry of the object must be done before refusing potential matches since the symmetry of an object can cause rejected matches. This is why the employment of RANSAC based homographies for spatial consistency only reduced the classification and recognition accuracy.

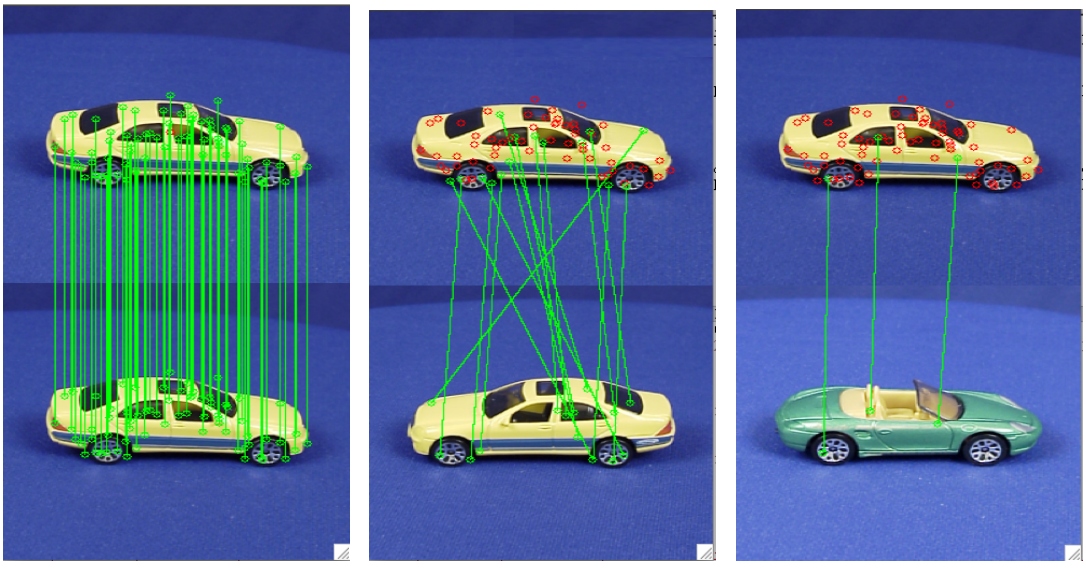


(a) SIFT feature self matches

(b) SIFT feature matching between two views of the same object. The object is recognized with 3 feature matches.

(c) SIFT feature matching between two instances of the Car object. Only the wheel is matched

Figure 75: SIFT descriptor performance examples

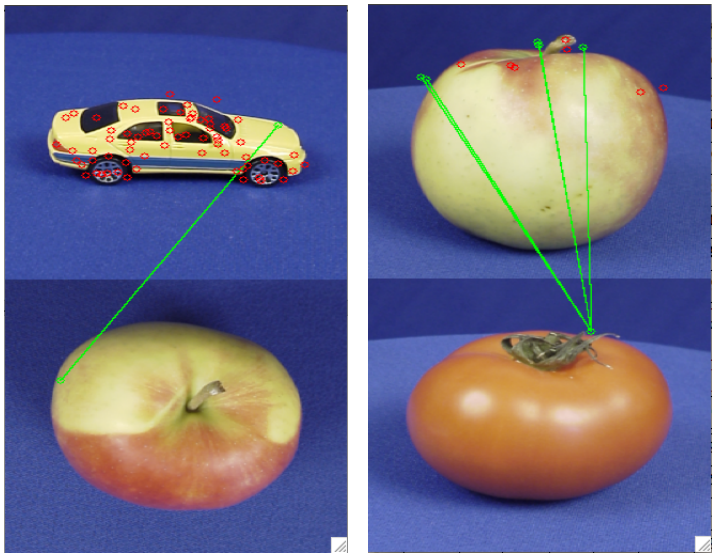


(a) SURF feature self-matches

(b) The SURF detector finds many more matches than SIFT on the recognition task.

(c) SURF matches 3 features in the this example, successfully classifying this as a Car.

Figure 76: SURF descriptor performance examples



(a) Here the SURF technique makes an incorrect match; however, the curvature of the edge at this point is very similar between these two objects.

(b) Here the SURF detector identifies multiple matches to one feature. We implemented a one-to-one limit for feature matching to prevent this.

Figure 77: SURF descriptor mistakes

	apple	car	cow	cup	dog	horse	pear	tomato
apple	**32**	6	0	17	1	3	4	9
car	6	**53**	2	4	1	0	6	0
cow	4	8	**30**	8	6	2	13	1
cup	4	1	1	**65**	0	0	0	1
dog	0	4	10	12	**33**	3	7	3
horse	7	5	2	10	1	**35**	10	2
pear	4	0	1	15	2	0	**46**	4
tomato	9	3	4	7	2	1	0	**46**

Table 16: Confusion matrix for recognition task, SIFT detector, no mask. Overall: 59% correct.

	apple	car	cow	cup	dog	horse	pear	tomato
apple	**32**	3	6	9	0	1	7	14
car	2	**54**	0	8	1	0	6	1
cow	4	3	**26**	14	7	1	11	6
cup	10	2	4	**55**	0	1	0	0
dog	3	4	8	13	**27**	1	6	10
horse	3	3	2	11	4	**31**	12	6
pear	5	6	7	13	1	2	**27**	11
tomato	4	1	6	12	1	1	6	**41**

Table 17: Confusion matrix for recognition task, SIFT detector, mask. Overall 51% correct.

	apple	car	cow	cup	dog	horse	pear	tomato
apple	**30**	5	5	21	1	5	5	8
car	11	**41**	3	8	2	1	9	5
cow	20	18	**20**	7	3	2	4	6
cup	13	12	7	**44**	0	0	2	2
dog	23	17	7	13	**6**	5	6	3
horse	18	16	12	8	5	**10**	9	2
pear	17	8	3	19	2	5	**18**	8
tomato	28	7	8	11	2	2	7	**15**

Table 18: Confusion matrix for recognition task, SIFT detector, mask with RANSAC step. Overall 29% correct.

	apple	car	cow	cup	dog	horse	pear	tomato
apple	**111**	21	10	181	31	36	67	119
car	40	**229**	40	114	25	18	54	56
cow	47	56	**148**	83	45	49	120	28
cup	83	53	17	**231**	36	8	92	56
dog	49	58	62	122	**114**	42	85	44
horse	53	62	71	106	48	**60**	134	42
pear	26	19	15	149	22	37	**255**	53
tomato	104	33	11	87	28	22	97	**194**

Table 19: Confusion matrix for classification task, SIFT detector, no mask. Overall 29% correct

	apple	car	cow	cup	dog	horse	pear	tomato
apple	**126**	17	44	130	18	15	93	133
car	39	**181**	30	151	18	24	57	76
cow	31	43	**128**	124	49	36	120	45
cup	114	40	27	**196**	39	15	95	50
dog	47	27	66	162	**83**	31	86	74
horse	38	50	65	142	41	**58**	124	58
pear	46	27	58	143	19	19	**155**	109
tomato	101	28	13	85	21	15	97	**216**

Table 20: Confusion matrix for classification task, SIFT detector, masked. Overall 25% correct.

	apple	car	cow	cup	dog	horse	pear	tomato
apple	**191**	49	51	211	18	53	52	87
car	140	**239**	54	110	28	29	37	75
cow	163	158	**146**	96	29	21	69	30
cup	242	87	43	**172**	35	12	72	49
dog	185	152	87	128	**31**	36	41	52
horse	163	162	110	107	30	**38**	62	40
pear	144	60	55	167	12	91	**94**	89
tomato	279	67	19	98	28	28	75	**118**

Table 21: Confusion matrix for classification task, SIFT detector, masked with RANSAC step. Overall 18% correct.

	apple	car	cow	cup	dog	horse	pear	tomato
apple	**58**	0	1	0	0	0	19	2
car	4	**76**	0	0	0	0	0	0
cow	4	0	**53**	0	2	5	14	2
cup	1	0	0	**78**	0	0	1	0
dog	8	3	3	4	**54**	0	2	6
horse	8	2	2	1	1	**58**	7	1
pear	0	0	0	0	0	1	**79**	0
tomato	15	0	0	0	0	1	4	**60**

Table 22: Confusion matrix for recognition task, SURF detector, unmasked. Overall 81% correct.

	apple	car	cow	cup	dog	horse	pear	tomato
apple	**383**	2	9	35	3	13	194	73
car	208	**228**	10	42	19	24	118	63
cow	135	14	**187**	41	67	68	179	21
cup	75	6	5	**380**	14	19	196	17
dog	192	11	45	33	**181**	61	153	36
horse	151	23	74	36	87	**135**	180	26
pear	131	3	12	36	9	19	**488**	14
tomato	266	8	8	18	9	12	146	**245**

Table 23: Confusion matrix for classification task, SURF detector, unmasked. Overall 39% correct.

	apple	car	cow	cup	dog	horse	pear	tomato
apple	**55**	0	0	1	1	0	22	1
car	6	**71**	0	1	1	0	0	1
cow	9	1	**36**	3	5	2	21	3
cup	2	0	1	**74**	0	0	3	0
dog	9	1	3	5	**46**	4	11	1
horse	17	0	1	2	2	**41**	15	2
pear	8	2	0	2	0	2	**66**	0
tomato	10	0	0	2	0	1	7	**60**

Table 24: Confusion matrix for the recognition task, SURF detector, masked. Overall 70% correct.

	apple	car	cow	cup	dog	horse	pear	tomato
apple	**309**	1	13	51	9	12	258	59
car	228	**176**	10	56	28	42	123	49
cow	150	10	**121**	21	60	68	262	20
cup	91	21	9	**344**	10	14	218	5
dog	218	14	37	35	**106**	58	209	35
horse	198	6	49	18	73	**102**	230	36
pear	132	11	36	89	31	25	**373**	15
tomato	238	7	7	12	16	13	204	**215**

Table 25: Confusion matrix for the classification task, SURF detector, masked. Overall 31% correct.

8.1.5 Conclusion

The classification accuracy was reduced significantly by my experimental design due to the fact that only one object instance at a time was compared. Classification can benefit from finding the best match out of a set of instances, it does not need to match every single instance of that class to be matched. The experimental design forced the classification to be computed based on having only one example instead of the 9 examples that it could have used and then taken the best result. This shortcoming will likely result in both the SIFT and SURF classification performances being lowered by the same amount, so comparisons between them are still valid.

The boundary features are a major component of the recognition when these affine invariant feature detectors are subjected to object transformations out of SO3. This is likely due to the fact that features on the boundary of objects represent a two-dimensional silhouette of the object. As the object undergoes transformations from SO3, the silhouette undergoes locally affine transformations, which fall under the invariance of these feature descriptors. This data set, while simple in many respects such as object placement and the absence of background distraction, is actually quite difficult to classify since it exhibits full rotation of the objects under study. For a given declination on the view sphere, there are only 8 views evenly spaced around the azimuth direction – so each view is significantly different than all others at 45 degrees apart. These descriptors are only meant to be invariant across scale, translation, illumination, and planar rotation – that they have performed this well on this difficult data set which exhibits full SO3 transformations is quite encouraging.

The classifier used in this paper is the most primitive type possible to use on these applications. Given a more sophisticated machine learning algorithm such as Ada-Boost or SVM the classification accuracy could be increased.

8.2 Loopy belief propagation

Computing the marginals of an arbitrary probability model can be na ively accomplished by computing the joint distribution over the entire model, and then summing out all other variables to yield each marginal. This algorithm is impractical for all but the smallest of graphical models due to exponential complexity.

Loopy belief propagation is an algorithm for computing marginal distributions in generic graphical models. This algorithm is exact when applied to tree-structured graphical models; however, it generates approximate marginals in graphs which contain loops.

The belief propagation algorithm works by iteratively refining marginal estimates at each node, while computing *messages* which are passed to neighboring nodes. A *message* sent from node A to node B is intuitively described as node B's best estimate for the distribution on A. The distribution on a node is computed by multiplying together messages from each neighbor.

$$p(x_i) = \frac{1}{Z} \prod_{j \in Ne(i)} \psi_{j \to i}(x_i) \tag{38}$$

$$\psi_{i \to j}(x_j) = \sum_{x_i} \phi_{i,j}(x_i, x_j) \phi_i(x_i) \prod_{k \in Ne(i)} \psi_{k \to i}(x_i) \tag{39}$$

In equation 38, the marginal on node x_i is computed as the normalized product of all incoming messages ψ. These messages come from each neighbor of the node for which the marginal is being computed. A message is computed as the marginalization of the neighbor from the product of the current marginal on the neighbor and the factor on the edge linking them. This process iterates until all messages have converged, which typically occurs within a few iterations; however, there are classes of graphs which will not converge or will converge to the wrong solution. In practice, loopy belief propagation usually works very well.

References

A. Agarwal and B. Triggs. Hyperfeatures- multilevel local coding for visual recognition. *European Conference on Computer Vision*, 2006.

R. De Almeida and C. Melin. Exploration of unknown environments by a mobile robot. *Intelligent Autonomous Systems*, 2:715–725, 1989.

ARL. Army Research Lab Micro Autonomous Systems and Technology Collaborative Technology Alliance MAST CTA. *http://www.arl.army.mil/www/default.cfm?page=332*, 2006.

A. Aydemir, M. Göbelbecker, A. Pronobis, K. Sjöö, and P. Jensfelt. Plan-based object search and exploration using semantic spatial knowledge in the real world. In *Proc. of the European Conference on Mobile Robotics (ECMR'11), Orebro, Sweden*, 2011.

T. Bailey. Constrained initialisation for bearing-only SLAM. *IEEE International Conference on Robotics and Automation*, 2003.

T. Bailey and H. Durrant-Whyte. Simultaneous localisation and mapping (SLAM): Part II state of the art. *Robotics and Automation Magazine*, September 2006.

D. H. Ballard. Generalizing the Hough transform to detect arbitrary shapes. In *Pattern Recognition*, 1981.

H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *European Conference on Computer Vision (ECCV)*, 2006.

M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mosenlechner, D. Pangercic, T. Ruhr, and M. Tenorth. Robotic roommates making pancakes. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 529–536. IEEE, 2011.

D. Benedettelli, A. Garulli, and A. Giannitrapani. Cooperative SLAM using M-Space representation of linear features. *Robotics and Autonomous Systems*, 60:1267–1278, 2012.

S. A. Berrabah, Y. Baudoin, and H. Sahli. SLAM for robotic assistance to fire-fighting services. *World Congress on Intelligent Control and Automation*, 2010.

C. Bibby and I. Reid. Simultaneous localisation and mapping in dynamic environments (SLAMIDE) with reversible data association. In *Robotics: Science and Systems*, 2007.

I. Biederman. Recognition-by-components: a theory of human image understanding. In *Psychological Review*, 1987.

O. Booij, B. Terwijn, Z. Zivkovic, and B. Krose. Navigation using an appearance based topological map. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3927–3932. IEEE, 2007.

M. Bosse, P. Newman, J. Leonard, and S. Teller. An *Atlas* framework for scalable mapping. *International Conference on Robotics and Automation*, 2003.

G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc., 2008.

W. Burgard, C. Stachniss, G. Grisetti, and B. Steder. Trajectory-based comparison of SLAM algorithms. *IEEE International Conference on Intelligent Robots and Systems*, 2009.

J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pages 679–698, 1986.

R. O. Castle, D. J. Gawley, G. Klein, and D.W. Murray. Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras. *International Conference on Robotics and Automation*, 2007.

A. Censi. An accurate closed-form estimate of ICP's covariance. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3167–3172. IEEE, 2007.

A. Censi. An ICP variant using a point-to-line metric. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 19–25. IEEE, 2008.

R. Chatila and J.P. Laumond. Position referencing and consistent world modeling for mobile robots. *International Conference on Robotics and Automation*, 1985.

R. Chipalkatty, H. Daepp, M. Egerstedt, and W. Book. Human-in-the-loop: Mpc for shared control of a quadruped rescue robot. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4556–4561. IEEE, 2011.

H. Choset and J. Burdick. Sensor-based exploration: The hierachical generalized voronoi graph. *The International Journal of Robotics Research*, 19(2):96–125, February 2000.

M. Ciocarlie, K. Hsiao, E. G. Jones, S. Chitta, R. B. Rusu, and I. A. Sucan. Towards reliable grasping and manipulation in household environments. In *International Symposium on Experimental Robotics (ISER)*, New Delhi, India, 12/2010 2010.

L. A. Clemente, A. J. Davison, I. Reid, J. Neira, and J. D. Tardos. Mapping large loops with a single hand-held camera. *Robotics: Science and Systems*, 2007.

J. Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. *International Conference on Robotics and Automation*, 1989.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2005a.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, page 886. Citeseer, 2005b.

A. J. Davison. *Mobile Robot Navigation Using Active Vision*. Ph.d. thesis, University of Oxford, 1998.

A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. *International Conference on Computer Vision*, 2003.

A. J. Davison and D. W. Murray. Simultaneous localisation and map-building using active vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

A. J. Davison, I. Reid, N. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

A. de la Escalera, J. María-Armingol, J. Manuel-Pastor, and F. José-Rodríguez. Visual sign information extraction and identification by deformable models for intelligent vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 5(2), June 2004.

F. Dellaert. Square root SAM: Simultaneous localization and mapping via square root information smoothing. In *Robotics: Science and Systems (RSS)*, 2005.

G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Robotic exploration as graph construction. *Transactions on Robotics and Automation*, 7(6):859–865, December 1991.

H. Durrant-Whyte and T. Bailey. Simultaneous localisation and mapping (SLAM): Part I the essential algorithms. *Robotics and Automation Magazine*, June 2006.

H. Durrant-Whyte and M. Stevens. Data fusion in decentralized sensing networks. In *4th Intl. Conf. on Information Fusion*, Montreal, 2001.

P. Espinace, T. Kollar, A. Soto, and N. Roy. Indoor scene recognition through object detection. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1406–1413. IEEE, 2010.

C. Estrada, J. Neira, and J. D. Tardos. Hierarchical SLAM: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4):588–596, August 2005.

G. J. Ettinger. Large hierarchical object recognition using libraries of parameterized model sub-parts. *AITR*, 1987.

L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In *Computer Vision and Pattern Recognition (CVPR)*, 2004.

R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 264–271, June 2003. URL `http://www.robots.ox.ac.uk/~vgg`.

S. Fidler and A. Leonardis. Towards scalable representations of object categories: Learning a hierarchy of parts. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

S. Fidler, G. Berginc, and A. Leonardis. Hierarchical statistical learning of generic parts of object structure. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

J. Folkesson and H. Christensen. Graphical SLAM - a self-correcting map. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

J. Folkesson, P. Jensfelt, and H. Christensen. Vision SLAM in the measurement subspace. *IEEE International Conference on Robotics and Automation*, 2005a.

J. Folkesson, P. Jensfelt, and H.I. Christensen. Graphical SLAM using vision and the measurement subspace. *IEEE International Conference on Intelligent Robots and Systems*, 2005b.

J. Folkesson, P. Jensfelt, and H. I. Christensen. The M-space feature representation for SLAM. *IEEE Transactions on Robotics*, 2007.

D. A. Forsyth and J. Ponce. *Computer Vision: A modern approach*. Pearson Education Inc, 2003.

D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart. Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, 94(7):1325–1339, 2006.

W. Freeman and E. Adelson. The design and use of steerable filters. *Pattern Analysis and Machine Intelligence*, 1991.

P. Gärdenfors. *Conceptual Spaces: The Geometry of Thought*. MIT Press, 2000.

I. Goodfellow, N. Koenig, M. Muja, C. Pantofaru, A. Sorokin, and L. Takayama. Help Me Help You: Interfaces for personal robots. In *Proc. of Human Robot Interaction (HRI)*, Osaka, Japan, 2010. ACM Press, ACM Press.

L. Van Gool, T. Moons, and D. Ungureanu. Affine/ photometric invariants for planar intensity patterns. *European Conference on Computer Vision*, 1996.

G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Efficient estimation of accurate maximum likelihood maps in 3d. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3472–3478. IEEE, 2007.

J. E. Guivant and E. M. Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation*, 17(3), June 2001.

D. Hähnel, D. Schulz, and W. Burgard. Map building with mobile robots in populated environments. *International Conference on Intelligent Robots and Systems*, pages 496–501, 2002.

D. Hähnel, R. Triebel, W. Burgard, and S. Thrun. Map building with mobile robots in dynamic environments. *International Conference on Robotics and Automation*, pages 1557–1563, 2003.

C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, 1988.

R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

F. Heger and S. Singh. Sliding autonomy for complex coordinated multi-robot tasks: Analysis and experiments. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

G. Hollinger, S. Singh, and A. Kehagias. Improving the efficiency of clearing with multi-agent teams. *The International Journal of Robotics Research*, 29(8):1088–1105, July 2010.

X. Hou and L. Zhang. Saliency detection: A spectral residual approach. In *Computer Vision and Pattern Recognition (CVPR)*, 2007.

M. Jason and S.M. LaValle. Localization with limited sensing. *IEEE Transations on Robotics*, 23:704–716, 2007.

S. Joyeux, R. Alami, S. Lacroix, and R. Philippsen. A plan manager for multi-robot systems. *International Journal of Robotics Research*, 28(2):220–240, 2009.

M. Kaess, A. Ranganathan, and F. Dellaert. Fast incremental square root information smoothing. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, (Forthcoming), 2008.

J. A. Kientz, S. N. Patel, B. Jones, E. Price, E. D. Mynatt, and G. D. Abowd. The georgia tech aware home. In *CHI '08 extended abstracts on Human factors in computing systems*, CHI EA '08, New York, NY, USA, 2008. ACM.

D. E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

M. Kochenderfer and R. Gupta. Common sense data acquisition for indoor mobile robots. In *National Conference on AI*, 2004.

J. Koenderink and A. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 1987.

T. Kollar and N. Roy. Utilizing object-object and object-scene context when planning to find things. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 2168–2173. IEEE, 2009.

S. Krempp, D. Geman, and Y. Amit. Sequential learning of reusable parts for object detection. *Tech Report at Johns Hopkins University*, 2002.

B. J. A. Kröse, N. Vlassis, R. Bunschoten, and Y. Motomura. A probabilistic model for appearance-based robot localization. *Image and Vision Computing*, 2001.

B. Kuipers. A hierarchy of qualitative representations for space. In *Spatial Cognition*, 1998.

B. J. Kuipers and Y. T. Byun. A qualitative approach to robot exploration and map-learning. In *IEEE Workshop on Spatial Reasoning and Multi-sensor Fusion*, pages 390–404, Los Altos, California, 1987.

B. Leibe and B. Schiele. Interleaved object categorization and segmentation. In *British Machine Vision Conference (BMVC)*, 2003.

D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004a.

D. G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision (ICCV)*, 1999.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision (IJCV)*, 2004b.

J. Ma. *Real-time Applications of 3D Object Detection and Tracking*. PhD thesis, Mechanical Engineering Department, California Institute of Technology, Pasadena, CA, June 2010.

S. Maldonaldo-Bascón, S. Lafuente-Arroyo, P. Gil-Jiménaz, H. Gómez-Moreno, and F. López-Ferreras. Road-sign detection and recognition based on support vector machines. *IEEE Transactions on Intelligent Transportation Systems*, 8(2), June 2007.

W. Meeussen, E. Marder-Eppstein, K. Watts, and B. P. Gerkey. Long term autonomy in office environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. *International Conference on Computer Vision*, pages 525–531, 2001.

K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. *European Conference on Computer Vision*, pages 128–142, 2002.

K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003.

S. Miller, M. Fritz, T. Darrell, and P. Abbeel. Parametrized shape models for clothing. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

J. M. M. Montiel, J. Civera, and A. J. Davison. Unified inverse depth parameterization for monocular SLAM. *Robotics: Science and Systems*, 2006.

O.M. Mozos, C. Stachniss, and W. Burgard. Supervised learning of places from range data using adaboost. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1730–1735. IEEE, 2005.

M. Müller, H. Surmann, K. Pervölz, and S. May. The accuracy of 6D SLAM using the AIS 3D laser scanner. *International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2006.

J. Neira and J.D. Tardós. Data association in stochastic mapping using the joint compatibility test. In *IEEE Transactions on Robotics and Automation*, 2001.

C. Neito-Granda, J. G. Rogers III, A. J. B. Trevor, and H. I. Christiensen. Semantic map partitioning in indoor environments using regional analysis. In *IEEE International Conference on Intelligent RObots and Systems (IROS)*, 2010.

H. Nguyen, A. Jain, C. Anderson, and C. Kemp. A clickable world: Behavior selection through pointing and context for mobile manipulation. In *IEEE International Conference on Intelligent RObots and Systems (IROS)*, 2008.

V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart. A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics. *International Conference on Intelligent Robots and Systems*, 2005.

K. Ni, D. Steedly, and F. Dellaert. Tectonic SAM: Exact; out-of-core; submap-based SLAM. In *International Conference on Robotics and Automation*, 2007.

JM O'Kane and SM LaValle. Almost-sensorless localization. *IEEE International Conference on Robotics and Automation*, pages 3764–3769, 2005.

A. Oliva and A. Torralba. The role of context in object recognition. In *Trends in cognitive science*, 2008.

E. Olson, J. Strom, R. Morton, A. Richardson, P. Ranganathan, R. Goeddel, M. Bulic, J. Crossman, and B. Marinier. Progress towards multi-robot reconnaissance and the MAGIC 2010 competition. *Journal of Field Robotics*, 29(5):762–792, September 2012.

A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. *European Conference on Computer Vision*, pages 575–588, 2006.

L. Parker. *Handbook of Robotics*, chapter Multiple Mobile Robot Systems, pages 921–941. Springer Verlag, New York/Berlin, May 2008.

J. Pineau and S. Thrun. High-level robot behavior control using POMDPs. In *AAAI-02 Workshop on Cognitive Robotics*, volume 107, 2002.

M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *International Conference on Computer Vision (ICCV)*, 2007.

J. Ramos. Using tf-idf to determine word relevance in document queries. In *Proceedings of the First Instructional Conference on Machine Learning*, 2003.

J. Rogers, A.J.B. Trevor, C. Nieto, A. Cunningham, M. Paluri, N. Michael, F. Dellaert, H.I. Christensen, and V. Kumar. Effects of sensory perception on mobile robot localization and mapping. In *International Symposium on Experimental Robotics (ISER)*, 2010.

J. Rogers, M. Paluri, A. Cunningham, H. I. Christensen, N. Michael, V. Kumar, J. Ma, and L. Matthies. Distributed autonomous mapping of indoor environments. In *SPIE Defense and Security*, 2011.

J.G. Rogers and H.I. Christensen. Normalized graph cuts for visual SLAM. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 918–923. IEEE, 2009.

J. Rogers III, E. Stump, S. Young, L. Sadler, and H. I. Christensen. Autonomous 3d exploration and mapping with unmanned ground robots. In *SPIE Defense, Security and Sensing*, Baltimore, April 2012a.

J. G. Rogers III and H. I. Christensen. A conditional random field model for place and object classification. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

J. G. Rogers III, A. J. B. Trevor, C. Nieto-Granda, and H. I. Christensen. SLAM with expectation maximization for moveable object tracking. In *IEEE International Conference on Intelligent RObots and Systems (IROS)*, 2010.

J. G. Rogers III, A. J. B. Trevor, C. Nieto-Granda, and H. I. Christensen. Simultaneous localization and mapping with learned object recognition and semantic data association. In *IEEE International Conference on Intelligent RObots and Systems (IROS)*, 2011.

J. G. Rogers III, C. Nieto-Granda, and Christensen H. I. Coordination strategies for multi-robot exploration and mapping. In *International Symposium on Experimental Robotics (ISER)*, 2012b.

N. Roy, G.J. Gordon, and S. Thrun. Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research*, 23:1–40, 2005.

D. Rus, B. Donald, and J. Jennings. Moving furniture with teams of autonomous robots. In *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 1, pages 235–242. IEEE, 1995.

R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011 2011.

F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets. *European Conference on Computer Vision*, 2002.

M. Schmidt. Undirected graphical model. In *http://www.di.ens.fr/ mschmidt/Software/UGM.html*, 2011.

A. Segal, D. Haehnel, and S. Thrun. Generalized-icp. In *Proc. of Robotics: Science and Systems (RSS)*, 2009.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision (ECCV)*, 2006.

J. Sivic and A. Zisserman. Efficient visual content retrieval and mining in videos. *Advances in Multimedia Information Processing-PCM 2004*, pages 471–478, 2005.

R. Smith. An overview of the tesseract OCR engine. *Proceedings of the ICDAR*, 2007.

R. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research (IJRR)*, 5(4):56–68, 1987.

R.C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research*, 5(4):56–68, 1986.

C. Stachniss and W. Burgard. Mobile robot mapping and localization in non-static environments. *Proceedings of the National Conference on Artificial Intelligence*, 20 (3), 2005.

C. Stachniss, O.M. Mozos, and W. Burgard. Speeding-up multi-robot exploration by considering semantic place information. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1692–1697. IEEE, 2006.

E. B. Sudderth, A. Torralba, W. T. Freeman, and A. S. Willsky. Learning hierarchical models of scenes, objects, and parts. *International Conference on Computer Vision*, 2005.

J. D. Tardos, J. Neira, P. Newman, and J. Leonard. Robust mapping and localization in indoor environments using sonar data. *International Journal of Robotics Research*, 2002.

A. Thayananthan, R. Navaratnam, B. Stenger, P. Torr, and R. Cipolla. Multivariate relevance vector machines for tracking. In *European Conference on Computer Vision*, pages 124–138. Springer-Verlag, 2005.

M. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

M. E. Tipping. The relevance vector machine. In *Advances in Neural Information Processing Systems 12 (NIPS)*, pages 652–658, 1999.

M. Tomono and S. Yuta. Mobile robot navigation in indoor environments using object and character recognition. In *International Conference on Robotics and Automation*, 2000.

M. Tomono and S. Yuta. Object-based localization and mapping using loop constraints and geometric prior knowledge. In *International Conference on Robotics and Automation*, 2003.

A. J. B. Trevor, J. G. Rogers III, C. Nieto-Granda, and H. I. Christensen. Applying domain knowledge to SLAM using virtual measurements. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

A. J. B. Trevor, J. G. Rogers III, and Christensen H. I. Planar surface SLAM with 3D and 2D sensors. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

E. Trucco and A. Verri. *Introductory techniques for 3-D computer vision.* Prentice Hall, 1998.

M. A. Turk and A. P. Pentland. Face recognition using Eigenfaces. In *Computer Vision and Pattern Recognition (CVPR)*, 1991.

M.M. Ullah, A. Pronobis, B. Caputo, J. Luo, R. Jensfelt, and H.I. Christensen. Towards robust place recognition for robot localization. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 530–537. IEEE, 2008.

R. Vincent, D. Fox, J. Ko, K. Konolige, B. Limketkai, B. Morisset, C. Ortiz, D. Schulz, and B. Stewart. Distributed multirobot exploration, mapping, and task allocation. *Annals of Mathematics and Artificial Intelligence*, 52(2):229–255, 2008.

P. Viola and M. J. Jones. Robust real-time face detection. In *International Journal of Computer Vision (IJCV)*, 2004.

D. Wagner and D. Schmalstieg. ARToolKitPlus for pose tracking on mobile devices. *Proceedings of the 12th Computer Vision Winter Workshop*, 2007.

C. Wang and C. Thorpe. Simultaneous localization and mapping with detection and tracking of moving objects. *International Conference on Robotics and Automation*, 2002.

C. Wang, C. Thorpe, and S. Thrun. Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. *International Conference on Robotics and Automation*, 2003.

O. Williams, A. Blake, and R. Cipolla. A sparse probabilistic learning algorithm for real-time tracking. In *Internation Conference on Computer Vision*, pages 353–360, 2003.

D. Wolf and G. S. Sukhatme. Towards mapping dynamic environments. *International Conference on Advanced Robotics*, 2003.

D. Wolf and G. S. Sukhatme. Online simultaneous localization and mapping in dynamic environments. *International Conference on Robotics and Automation*, 2004.

D. Wolf and G. S. Sukhatme. Mobile robot simultaneous localization and mapping in dynamic environments. *Autonomous Robots*, 2005.

B. Yamauchi. A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 146–151. IEEE, 1997.

B. Yamauchi, A. Schultz, and W. Adams. Integrating exploration, localization, and navigation. *Adaptive Systems*, 7(2):75–82, 1999.

H. Zender, O. Martínez Mozos, P. Jensfelt, G.J.M. Kruijff, and W. Burgard. Conceptual spatial representations for indoor mobile robots. *Robotics and Autonomous Systems*, 56(6):493–502, 2008.

K. Zheng, D.F. Glas, T. Kanda, H. Ishiguro, and N. Hagita. How many social robots can one operator control? In *Proceedings of the 6th international conference on Human-robot interaction*, pages 379–386. ACM, 2011.